



FRIEDRICH-SCHILLER-UNIVERSITÄT JENA
FAKULTÄT FÜR MATHEMATIK UND INFORMATIK
INSTITUT FÜR INFORMATIK

Ontologiestützte Websuche -
auf Grundlage des Google Web APIs Services

DIPLOMARBEIT

zur Erlangung des akademischen Grades
Diplom-Informatiker

eingereicht von UWE KRÜGER
geb. am 29. Mai 1972 in Eisenberg

Betreuer: Prof. Dr. MARTIN MUNDHENK
Jena, 21. Juli 2005

Abstract

Das World Wide Web ist trotz seiner heutigen enormen Bedeutung noch lange nicht am Ende seiner Entwicklung angelangt. Sein exponentielles Wachstum erfordert das Erforschen neuer Ansätze, um der wachsenden Informationsflut Herr zu werden. Die Realisierung der schon seit den 90er Jahren existierenden *Semantic Web* Vision, stellt dabei eine der spannendsten und größten Herausforderungen der zukünftigen Webentwicklung dar.

Bis ein semantisches Netz etabliert ist, gilt es noch viele Hürden zu überwinden. Jedoch stehen die grundlegenden Technologien wie *XML*, *RDF*, *OWL*, *Web Services* etc. heute schon bereit. In dieser Arbeit wird untersucht, wie es durch Einsatz dieser Technologien bereits heute möglich ist, bei einer konventionellen schlüsselwortbasierten Websuche, zu den Treffern einer Suchanfrage zusätzliche *Semantik* anzuzeigen. Dem Nutzer soll es ermöglicht werden, navigierend mit Hilfe der zusätzlichen Bedeutungszuordnung seine Suche effektiv einzugrenzen, um somit die Vielzahl von Suchtreffern auf relevante Treffer einzuschränken. Umgesetzt wurde dieser Ansatz für das Webseiten-Angebot der Domäne der Friedrich-Schiller-Universität Jena (*fsu-jena.de*). Die semantische Information der Webseitenstruktur der FSU-Jena wurde zuerst in einer eigens erstellten *Ontologie* unter Verwendung des *OWL*-Sprachstandards modelliert. Als Datengrundlage für die Websuche kommt der *Google API Web Service*, unter Verwendung des Webservice-Kommunikations-Protokolls *SOAP*, zum Einsatz. Nach Erläuterung der theoretisch technischen Grundlagen der eingesetzten Technologien beschreibt diese Arbeit, auf welchem Weg und inwieweit es gelungen ist, die in der Ontologie modellierte Wissensbasis mit den Ergebnissen der Suchmaschine, in einer eigens programmierten Web-Anwendung zu verbinden. Die im Zuge der Umsetzung aufgedeckten Probleme werden dabei an gegebener Stelle jeweils kurz kritisch diskutiert.

Inhaltsverzeichnis

Abstract	II
Tabellenverzeichnis	VI
Abbildungsverzeichnis	VII
Listingsverzeichnis	VIII
Abkürzungen und Akronyme	IX

Einleitung	1
-------------------	----------

I Grundlagen	4
---------------------	----------

1 Das World Wide Web	4
-----------------------------	----------

2 Suchdienste im World Wide Web	5
2.1 Grundtypen von Suchdiensten	6
2.2 Funktionsweise von Suchmaschinen	7
2.3 Grenzen heutiger Suchdienste	7
2.4 Herausforderung an die Websuche	9

3 Semantic Web	11
3.1 Die Vision eines semantischen Netzes	11
3.2 Die Semantic Web Architektur	14
3.2.1 Unicode + URI	14
3.2.2 Extensible Markup Language	14
3.2.3 Resource Description Framework	15
3.2.4 Ontologien (Ontology vocabular)	18
3.2.5 Wissensverarbeitung (Logic)	18
3.2.6 Automatische Beweisführung (Proof)	19
3.2.7 Vertrauen/Sicherheit (Trust)	19

4 Wissensbeschreibung durch Ontologien	20
4.1 Die Web Ontology Language	21
4.1.1 OWL Sprachebenen	22
4.1.2 Aufbau einer OWL-Datei	23
4.2 Ontologie-Editoren	26
4.2.1 Das Protégé-Projekt	26
4.2.2 Plugins für Protégé	27

5	Web Services	28
5.1	Das Web Service-Protokoll SOAP	30
5.1.1	Übermittlung der Nachricht	30
5.1.2	Aufbau der Nachricht	31
5.1.3	Inhalt der Nachricht	32
5.1.4	Transportprotokolle	32
5.2	Googles Web Service	33
5.2.1	Funktionalitäten	33
5.2.2	Nutzungsbedingungen und Einschränkungen	34
5.2.3	SOAP-Nachrichtenaustausch	35
II	Implementierung	37
6	Vorgehensweise	37
7	Beziehen der Datengrundlage	38
7.1	Methode des Screen Scrapings	39
7.2	Nutzung des Googles Web Services	40
7.3	Anfragesteuerung in einer eigenen Anfrage-Klasse	44
8	Erstellen der Ontologie	45
8.1	Festlegen der benötigten Klassen	46
8.2	Definition der benötigten Eigenschaften	47
8.3	Wissensakquise – Das Schaffen einer Wissensbasis	48
8.4	Ontologiepflege	50
8.5	Anforderungen an eine <i>Sonto</i> ^x -konforme Ontologie	50
9	Vorverarbeitung der Ontologie	51
9.1	Entwicklung eines eigenen OWL-Parsers	52
9.2	Erstellung einer OWL Parser-Klasse	53
9.3	Speicherung des Parserergebnisses	54
10	Verbindung der Ontologie mit der Datengrundlage einer Suchmaschine	55
10.1	Zuordnung der Suchtreffer zu den Individuen	58
10.2	Einschränkung des Suchraumes	59
10.2.1	Multiple Webresource-URLs	60
10.2.2	Behandlung multipler URLs und der Verzeichnisstrukturen	61
10.2.3	Zusätzliches Problem mit den Verzeichnisstrukturen	61
10.2.4	Probleme mit der URL-Struktur	63
10.3	Erweitertes Information Retrieval	63
10.4	Hilfestellung auf Basis der Ontologie für die Suchraumeinschränkung	65
10.5	Unschärfe Suche zur Suchraumerweiterung	67
10.6	Zusatzinformationen zur aktuellen Suchraumeinschränkung	69

III Auswertung und Zusammenfassung	71
11 Betrachtung der Umsetzung im Hinblick auf die Problemstellung	71
11.1 Stärken von <i>Sonto</i> ^X	71
11.2 Grenzen von <i>Sonto</i> ^X	72
12 Einsatzszenario des <i>Sonto</i>^X-Systems	73
13 Stellung von <i>Sonto</i>^X in der Semantic Web Vision	74
14 Zusammenfassung	75
15 Ausblick	76
A Glossar	77
B Das Architektur-Modell von <i>Sonto</i>^X	79
C Screenshots des <i>Sonto</i>^X Web-Interfaces	80
Literaturverzeichnis	82

Tabellenverzeichnis

1	Parameter für die Google Anfrage	35
2	Definierte Eigenschaften der Ontologie.	47
3	Parser-Ausführungszeit.	55
4	Implementierte Methoden der CONTROL-Klasse	57

Abbildungsverzeichnis

1	Teilausschnitt der Metasuchmaschine Kartoo a) und des TouchGraph GoogleBrowsers b)	10
2	Schichtenmodell der Semantic Web Architektur	14
3	Das RDF-Dreigespann	16
4	RDF-Beziehungen als Graph (N3-Notation)	17
5	Kommunikationssituation – Semiotisches Dreieck	20
6	OWL Sprachebenen	22
7	Das GUI von Protégé V3.0	27
8	Hierarchiedarstellung mit dem Plugin Jambalaya.	28
9	Schema der service-orientierten Web Service-Architektur	29
10	Aufbau einer SOAP-Nachricht.	31
11	Architektur-Modell mit gekennzeichneteter Schnittstelle zur Datenbeschaffung.	39
12	Google API Service als Datengrundlage	41
13	Das Klassen-Konzept der Ontologie.	46
14	Themenkuster der Webseitenstruktur der FSU Jena.	49
15	Verzeichnis- und Dateistruktur des <i>Sonto</i> ^x -Systems.	56
16	Beispielzuordnung der Individuen zu den Treffern.	58
17	URL-Struktur anhand eines Beispiels	59
18	Alternativer IR-Ansatz	64
19	Beispiel einer generierten Taxonomie.	66
20	Beispiel einer zugrunde liegenden Struktur für eine Taxonomie.	66
21	Auszug aus der Taxonomie.	67
22	Auszug aus der Taxonomie.	68
23	Zusätzliche Informationen zu der aktuellen Suchraumeinschränkung. . . .	69
24	Web-Interface: Startseite (http://www.artusweb.de/SontoX/index.html) . .	80
25	Web-Interface: Erweiterte Einstellungen (adv_search.php5)	80
26	Web-Interface: Ergebnis einer Beispielanfrage (search.php5)	81

Listings

1	Quelltext-Beispiel einer HTML-Webseite	12
2	Beispiel eines Statements in XML-Syntax	17
3	XML-Namensraum Deklaration	23
4	OWL-Header Definition	23
5	OWL-Class Definition	24
6	OWL-Property Definition	24
7	OWL-Individual Definition	25
8	Beispiel einer einfachen SOAP-Nachricht.	31
9	Auszug aus der WSDL-Datei.	34
10	<i>doGoogleSearch</i> (SOAP-Request)	35
11	<i>doGoogleSearch</i> (SOAP-Response)	36
12	SOAP-Client unter Verwendung der NUSOAP-Klasse.	42
13	Parameter für <i>doGoogleSearch</i> (WSDL-Datei).	42
14	Datentypen <i>GoogleSearchResult</i> und <i>ResultElement</i> (WSDL-Datei). . . .	43
15	Verarbeiten der Suchergebnisse in der INQUIRY-Klasse.	44
16	Auszug aus der Klassendefinition (<i>fsu-jena.owl</i>).	46
17	Auszug der Definition für <i>gehört_zu</i> und <i>Homepage</i> (<i>fsu-jena.owl</i>). . . .	48
18	Auszug aus der Individuen-Definition (<i>fsu-jena.owl</i>).	49
19	Konstruktor der OWLP-Klasse	53
20	Struktur des Individuum-Arrays.	54
21	Aufruf von <i>mapHomepage()</i> in <i>getResult()</i>	59
22	Festlegen der Reihenfolge für die Informationsanzeige (<i>config.php</i>)	69
23	Beispiel des Quelltextes für das Einbetten der Anzeige der Zusatzinfor- mationen.	69

Abkürzungen und Akronyme

API	Application Program Interface
ASCII	American Standard Code for Information Interchange
CGI	Common Gateway Interfaces
DNS	Domain Name Service
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
OWL	Web Ontology Language
PHP	PHP Hypertext Preprocessor
RDF	Resource Description Framework
RDFS	RDF Schema
RPC	Remote Procedure Calls
SMTP	Simple Mail Transport Protocol
SQL	Structured Query Language
SW	Semantic Web
TCP	Transmission Control Protocol
UDDI	Universal Description, Discovery and Integration (of Web Services)
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
W3C	World Wide Web Consortium
WSDL	Web Services Description Language
WWW	World Wide Web
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language
XMLS	XML Schema
XSL	Extensible Style Language
XSLT	XSL Transformations

Einleitung

Die Entwicklung des *World Wide Web*¹ (WWW) hat heute ein Stadium erreicht, in dem sich die Suchdienste einer immer schneller wachsenden Flut an Dokumenten gegenüber sehen. Eine reine, schlüsselwortbasierte Volltextsuche einer Suchmaschine kann dem Nutzer nur ein Suchergebnis auf Basis der Schlüsselwortvergleiche liefern. Eine inhaltliche Einordnung einer bestimmten Webseite zu einem Bedeutungskreis ist aufgrund dieser Strategie kaum möglich. Die von einem Suchdienst pro Suchtreffer angegebenen Informationen, wie der Titel, ein kurzer Textabriss (Engl. snippets) und einen URL reichen dabei meist nicht aus, um eine adäquate Relevanzprüfung auf einen Blick zu ermöglichen. So muss der Nutzer die gefundenen Suchtreffer manuell auf ihre Relevanz hin überprüfen, was unter Umständen einen erheblichen Aufwand darstellt, wobei der Erfolg nicht einmal garantiert ist. Dies spiegelt die im Allgemeinen heute vorherrschende Vorgehensweise bei einer Webrecherche wider.

In dieser Arbeit soll geklärt werden, welche Möglichkeiten es gibt, dem Nutzer schon im Vorfeld Hinweise über die inhaltliche Einordnung der einzelnen Suchtreffer einer Volltextsuchmaschine zu präsentieren. Der Nutzer soll möglichst auf den ersten Blick erkennen, welche Treffer für ihn eine hohe Relevanz darstellen und welche hingegen uninteressant sind. Erreicht werden soll dies durch die Darbietung zusätzlicher Informationen, die dem Nutzer eine navigierend sukzessive Verfeinerung der Suche ermöglicht. Eine Schlüsselrolle soll hierfür der Einsatz einer formalen Wissensbeschreibung – einer sog. *Ontologie* – einnehmen. Konkret besteht das Ziel darin, beispielhaft eine ontologiegestützte Websuche für das Webangebot der Friedrich-Schiller-Universität Jena zu implementieren. Auf Basis der Datengrundlage einer Suchmaschine soll dem Nutzer damit eine speziell erweiterte Websuche geboten werden, die zusätzlich die Suchtreffer visuell den einzelnen Bereichen der Universitätsstruktur zuordnet und bei Bedarf eine Möglichkeit für eine Sucheinschränkung bietet. Weiterhin soll untersucht werden, wie eine Anreicherung an Semantik zu den Suchtreffern erreicht werden kann.

Für obige Zielsetzung ist es notwendig, einen Überblick über den heutigen Entwicklungsstand des WWW und den darin zur Verfügung stehenden Suchdiensten zu erhalten. Die Möglichkeiten einer Ontologierealisierung müssen im Hinblick auf Kodierungsstandards und konzeptuelle Modellierung geprüft werden. Die konkrete Umsetzung einer Ontologie erfordert dabei ein hohes Maß an Rechercharbeit, um die für die Aufgabenstellung relevanten Universitätsstrukturen zu identifizieren und zu formalisieren. Weiterhin muss eine möglichst stabile Lösung für die Erhaltung einer Datengrundlage, auf der die erweiterte Suche aufsetzen kann, gefunden werden. Hierzu sind eine Betrachtung der prinzipiell möglichen Varianten und deren Einsatztauglichkeit notwendig. Flexibilität und Stabilität sollen hierfür zwei wichtige Entscheidungskriterien darstellen. Die zentrale Herausforderung dieser Arbeit ist die Verbindung von Ontologie und Suchmaschine. Folgende Frage-

¹ Auf den folgenden Seiten wird für den Begriff „World Wide Web“, je nach Kontext, synonym auch *Netz* oder *Web* verwendet.

stellung steht dabei im Mittelpunkt: Wie kann das in der Ontologie formal gefasste Wissen über die Universitätsstruktur für eine semantische Aufbereitung des Suchergebnisses einer „konventionellen“ Suchmaschine eingesetzt werden?

Obwohl die zugrunde liegenden Technologien schon seit einigen Jahren zur Verfügung stehen, betritt eine mögliche Umsetzung der Anforderungen weitgehend Neuland. Theoretisch ist der Wunsch nach einer semantischen Kennzeichnung der Suchtreffer schon einige Zeit im Gespräch, konkrete Umsetzungen auf diesem Gebiet sind jedoch kaum anzutreffen und wenn, dann stecken sie noch in den Kinderschuhen oder stellen nur eine spezielle Teillösung dar. Es gilt daher, eine eigene, neue und kreative Lösung für eine mögliche Ontologieintegration zu finden und zu implementieren. Dabei soll es ausreichend sein, eine „Insellösung“, zugeschnitten auf das Webseitenangebot der Friedrich-Schiller-Universität Jena, zu implementieren, anhand derer die universelle Einsetzbarkeit der gefundenen Lösung beispielhaft demonstriert wird.

Aufbau der Arbeit

Die theoretischen Grundlagen der behandelten Themenbereiche werden zu Beginn im Teil I vorgestellt. Eine kurze Einführung zur Entwicklung des WWW bis zum heutigen Tage bildet die Basis für ein Verständnis der Richtung der angestrebten Weiterentwicklung des WWW. Die Schwierigkeiten bzw. Aufgaben, die es mit dem heutigen Entwicklungsstand des WWW zu bewältigen gilt, werden in Bezug auf die Arbeit kurz angeschnitten. Darauf folgend werden vorhandene Konzepte von Suchdiensten aufgezeigt und deren prinzipielle technische Arbeitsweise erläutert. Daran anschließend wird in diesem Zusammenhang die Vision des Semantic Web vorgestellt. Dabei werden die einzelnen Etappen hin zur Erfüllung dieses Konzeptes kurz besprochen. Die nun geschaffene Grundlage führt im Anschluss auf eine genauere Betrachtung des zentralen Begriffes Ontologie und der sich dahinter verbergenden Relevanz für die Weiterentwicklung des WWW und für diese Arbeit. In diesem Zusammenhang wird als Entwicklungswerkzeug ein Ontologieeditor vorgestellt, der in dieser Arbeit zum Einsatz kam. Weiterhin wird die zur möglichen formalen Ausgestaltung einer Ontologie einsetzbare Ontologiesprache OWL vorgestellt. Ebenso wird, die für die Datengewinnung eingesetzte Technik der Web Services und als konkretes Beispiel der für die Implementierung favorisierte Google Web Service erläutert.

Eine Beschreibung der konkreten Vorgehensweise bezüglich der Umsetzung der Problemstellung, erfolgt im Teil II. Nach einer Begründung über die getroffene Wahl der für die Implementierung eingesetzten Programmiersprache wird als erster Schritt die Beschaffung einer Datengrundlage in Form einer automatisch gestellten Suchabfrage an eine Suchmaschine erläutert. Es folgt eine Beschreibung der vorgenommenen Schritte zur Ontologiemodellierung. Als weiteren Punkt wird anschließend das Problem der automatischen Ontologieverarbeitung, und dort konkret das Erstellen eines speziell zugeschnittenen Parsers aufgezeigt. Auf dieser Grundlage wird anhand einzelner Komponenten erläutert, an welchen Stellen der programmierten Websuche eine Integration der Ontologie zu den einzelnen Suchtreffern erreicht wurde und welche Techniken und Ideen dafür zum Einsatz kamen.

In Teil III wird zu Beginn eine Auswertung der erreichten Umsetzung anhand einer Analyse der Schwächen und Stärken vorgenommen. Die Funktionsweise der programmierten Websuche wird im Weiteren mit Hilfe zweier beispielhafter Suchszenarien überprüft und analysiert. Es folgt eine Bewertung der umgesetzten Websuche im Kontext der Semantic Web Vision. Eine Zusammenfassung, gefolgt von einer Betrachtung der zukünftigen Weiterentwicklung und Akzeptanz der bereitgestellten Websuche, schließt den Teil III ab.

Teil I

Grundlagen

1 Das World Wide Web

Begonnen hat alles Ende der 80er Jahre des vergangenen Jahrhunderts mit einer Idee von Tim Berners-Lee, der als Vater des WWW angesehen werden kann. Seine Vision bestand darin, beliebige Informationen so zu verknüpfen, dass dadurch ein globaler Informationsraum entsteht, der es ermöglichen sollte, bequem auf das Wissen der ganzen Menschheit zuzugreifen. Analog zu der uns Menschen gegebenen Fähigkeit, Assoziationen zwischen augenscheinlich nicht zusammenhängenden Dingen zu bilden, sollte es doch möglich sein, verschiedene Informationen oder Daten ebenfalls auf diese Art miteinander zu verknüpfen. Frei nach der Maxime „Das Ganze ist mehr als die Summe seiner Teile“ könnte ein solcher Informationsraum bis dahin ungeahnte Anwendungsgebiete entstehen lassen und Zusammenhänge aufdecken, die sonst unerkannt geblieben wären. Berners-Lees Vision war, dass potenziell alles mit allem verknüpft werden sollte, wodurch sich ein Netz aus Informationen bildet [Ber99].

Es ist bemerkenswert, dass diese futuristisch anmutende Idee aus den frühen 90er Jahren des vergangenen Jahrhunderts heute weitgehend Wirklichkeit geworden und sogar aus dem alltäglichen Leben nicht mehr wegzudenken ist.² Es ist heute kaum vorstellbar, dass die Entwicklung des WWW vor nicht einmal 15 Jahren ihren Anfang genommen hat. Natürlich war der Erfolg abhängig von der Schaffung eines weltweit umspannenden Computernetzwerkes, dem Internet, jedoch dürfte das enorme Wachstum des WWW wohl selbst Berners-Lee überrascht haben.

Der Erfolg des WWW beruht auf der Nutzung zweier relativ einfacher Standards, nämlich der *HyperText Markup Language (HTML)* zur Kodierung und dem *HyperText Transfer Protocol (HTTP)* zur Übertragung der Information. Darüber hinaus ist es jedem Nutzer möglich, beliebige Informationen weltweit abrufbar bereitzustellen. Je mehr von dieser Möglichkeit Gebrauch machten, umso mehr vergrößerte sich der Nutzen und die hierdurch bereitgestellte Informationsmenge, was wiederum zu einer breiteren Akzeptanz und somit zu einem neuen Entwicklungsschub führte. Schon im Sommer 1993 hatte sich die Entwicklung des WWW verselbständigt: „*I no longer had to push the bobsled. It was time to jump in and steer.*“ [Ber99]

Schätzungen zufolge ermöglicht das heutige WWW den Zugriff auf mehr als 50 Milliarden Webdokumente, wobei sich die Anzahl ca. alle 6 Monate verdoppelt ([Ber01, GS05]).

² Eine Bemerkung sei an dieser Stelle erlaubt: Der überwiegende Teil der Bevölkerung auf der Erde hat – zumeist aus ökonomischen Gründen – *keinen* Zugang zu der WWW-Technologie. Selbst in einem reichen Land wie den USA, hat ca. ein Drittel der Bevölkerung keinen Anteil an dem technologischen Fortschritt des WWW (Vgl. [Wei01] S. 15 ff.). Dies sollte nicht vergessen werden, wenn von dem *globalen* Siegeszug des WWW die Rede ist.

Will man jedoch heute die volle angebotene Informationsmenge nutzen, so gestaltet sich dies zunehmend schwieriger. Oft wird in diesem Zusammenhang von einem *Problem* des WWW, hervorgerufen durch eine inflationäre Situation des Informationsangebotes (Information-Overload), gesprochen. Hierbei von einem Problem zu sprechen, ist zwar inhaltlich nicht falsch, hinterlässt aber den Eindruck, dass das heutige WWW in sich fehlerhaft sei. Unbestritten hat das exponentielle Wachstum des WWW zur Folge, dass es für uns Menschen unmöglich geworden ist, alle Informationen zu erfassen oder die für einen selbst relevanten Informationen in adäquater Zeit überhaupt zu finden. Es muss jedoch beachtet werden, dass das WWW sich ständig weiterentwickelt und expandiert. Die Eigendynamik ist dabei die Triebkraft seines Erfolges. Die dadurch resultierenden – wenn auch momentan unbeherrschbaren – Informationsmengen stellen dabei kein eigentliches Problem des WWW dar, sondern sind vielmehr eine logische und auch gewünschte Konsequenz der Philosophie des Webs: „Jeder kann Informationen bereitstellen“.

Es gilt nun ein neues Kapitel in der WWW-Entwicklung zu öffnen und sich den Herausforderungen der neuen Gegebenheiten zu stellen. Die Aufgabe besteht in der Entwicklung neuer und in der Weiterentwicklung bereits bekannter Technologien, um der wachsenden Informationsflut Herr zu werden und damit die riesige Menge an Webdokumenten besser und auf eine neue Art nutzen zu können.

2 Suchdienste im World Wide Web

Wurde zu Beginn des WWW noch eine Art Liste der verfügbaren und neu hinzugekommenen Webseiten gepflegt, so waren aufgrund des rasanten Zuwachses die Grenzen des Darstellbaren bald erreicht [Ber99]. Bereits kurz nach der „Erfindung“ des WWW wurden Webseiten angeboten, die nur den Zweck dienten, dem Nutzer bei einer Webrecherche zu unterstützen. Zu den Pionieren dieser Idee gehören z. B. die Suchdienste *Yahoo* (<http://www.yahoo.com>) und *metacrawler* (<http://www.metacrawler.com>). Beide wurden 1994 gegründet und existieren, wenn auch in erweiterter Form, heute noch. Derzeit stehen unzählige verschiedenartige Suchdienste im WWW bereit und auch hier kommen fast täglich neue Vertreter hinzu. Die Suchdienste sind dabei ein fester Bestandteil bei der Nutzung des WWW geworden. Sie sind für viele Internetnutzer ein, wenn nicht DAS wichtigste Hilfsmittel bei einer Webrecherche und werden dabei gerne als erster „Webeinstieg“ genutzt. So einheitlich die Benutzerschnittstellen und die Intensionen der Suchdienste sind, so unterschiedlich sind meist die eingesetzten Mechanismen und Strategien, welche sie zur Erbringung ihres Services einsetzen.

Im Folgenden soll ein grober Überblick über die grundlegenden Konzepte und Eigenheiten der verschiedenen angebotenen Suchdienste gegeben werden. Eine umfassende Vorstellung der gesamten Funktionsweise der verschiedenen Suchstrategien soll im Rahmen dieser Arbeit nicht geboten werden. Hierzu sei, über die angebotene Literatur³ hinaus, auf die zahlreichen im Netz verfügbaren Ausarbeitungen, Spezifikationen und Dokumentationen zum Thema verwiesen.

³ Das Buch von Glöggler [Glö03] bietet hier z. B. einen ersten Einstieg.

2.1 Grundtypen von Suchdiensten

Wer im WWW recherchiert, hat heute die Qual der Wahl zwischen einer Vielzahl von Suchdiensten verschiedenster Anbieter. Um einen Überblick über die verschiedenen Arten der Datenbeschaffung, Aufbereitung und Darbietung zu erhalten, ist eine Einteilung der Suchdienste in drei Grundtypen empfehlenswert: ([Glö03] S. 1–11)

- ❑ **Suchmaschinen** bieten eine sog. *Volltextsuche*, auf einen zuvor automatisch erfassten und automatisch indizierten Seitenbestand (auch „indexbasierte Suche“ genannt), d. h. ein zuvor vom Nutzer eingegebenes Suchwort wird mit einer Schlüsselwortliste verglichen. Daraufhin werden die Links aller Webseiten, für die die gesuchten Schlüsselwörter als relevant eingestuft wurden, in einer nach Relevanz gestaffelten Rangordnung angezeigt. Ein prominenter Vertreter dieser Gattung ist z. B. der Google-Suchdienst (www.google.com).
- ❑ **Webkataloge** werden redaktionell gepflegt, d. h. letztendlich entscheiden Menschen darüber, *ob* und *wie* bestimmte Webseiten in den Katalog aufgenommen werden. Die einzelnen Seiten werden zuvor von einem Mitarbeiter begutachtet und dann – wenn kein Grund dagegen spricht – unter einer passenden Kategorie abgelegt. Dies hat zur Folge, dass unrelevante, unseriöse oder gar kriminelle Angebote von vornherein herausgefiltert werden, womit ein hohes Qualitätsniveau der Suchtreffer sichergestellt werden kann. Jedoch ist dies der Grund dafür, dass die Webkataloge nicht das vollständige Web berücksichtigen. Weiterhin fallen die Aktualisierungszyklen im Vergleich zu den Suchmaschinen länger aus. Als ein Vertreter dieses Typs sei der Webkatalog *dmoz* (www.dmoz.org) genannt.
- ❑ Das Konzept der **Meta-Suchmaschinen** verbindet gleichzeitige Suchanfragen an unterschiedliche Suchdienste, ob durch Webkatalog oder Suchmaschine, zu einem neuen gesamten und überarbeiteten Suchergebnis. Da die große Zahl der Ergebnisse jedoch zuvor einzeln von den verwendeten Suchmaschinen bzw. -katalogen abgefragt und aufbereitet werden muss, benötigt eine Suchanfrage naturgemäß eine längere Zeit. Sie deckt jedoch in der Regel einen größeren Seitenbestand ab und kann somit einen möglichst großen Teil der im Web verfügbaren Dokumente berücksichtigen.

Darüber hinaus existieren Suchdienste, die sich auf einen bestimmten Bereich fokussiert haben. Hierzu zählen spezielle *E-Commerce*-, *Multimedia*- und *Topic*-Suchdienste. Hinzu kommt die sog. *Deep Web*-Suche, bei der über eine Webschnittstelle in den Datenbeständen diverser Datenbanken gesucht werden kann.

An dieser Stelle soll kurz auf die sog. *Payed Placement*-Suchmaschinen eingegangen werden. Z. B. räumt der *Overture*⁴ Service, den Kunden gegen Bezahlung einen vorderen Platz in seinen Suchtreffern ein. Diese Vorgehensweise führt zu einer Mischung aus indexbasierten Suchtreffern und bewusst platzierten kommerziellen Einträgen. Nach Meinung des Autors liefert dieses Konzept eine recht exotische Treffermischung, die kaum eine objektive Repräsentation der im WWW bereitgestellten Informationen bietet und somit eher eine spezielle Rolle in der Suchdienstlandschaft des WWW einnimmt.

⁴ *Overture* ist eine Service der Yahoo-search-marketing Initiative. <http://www.overture.com>

2.2 Funktionsweise von Suchmaschinen

Da die Suchmaschinen ein wichtiges Fundament für die Websuche bilden, soll im folgenden Abschnitt deren allgemeine Funktionsweise grob erläutert werden. Traditionell lassen sich Suchmaschinen in nachstehende Komponenten untergliedern: (zusammenfassend aus [Glö03])

- ❑ Ausgehend von einer bereits bekannten Webseite werden sog. *Webrobot-Systeme*⁵ zur Analyse der Hyperlinks dieser Seite eingesetzt. Die Hyperlinks führen wiederum zu anderen Webseiten deren Linkstruktur nun ebenfalls analysiert wird. Das Ergebnis ist eine große Menge von URLs die in einer Datenbank abgelegt werden. Ein Webrobot ist demnach für die Erfassung von neuen und veränderten Ressourcen im Netz zuständig.
- ❑ Die *Indexing Software* bildet aus den registrierten Seiten eine Datenstruktur, die effizient durchsucht werden kann. Dazu müssen relevante Aspekte einer Webseite extrahiert werden. Die entsprechenden Informationen können beispielsweise Seiteninhalt, Metatags oder Hyperlinkstrukturen liefern. Die automatische Analyse und inhaltliche Bewertung erfolgt durch sog. *Information Retrieval Systeme* (IR-Systeme). Ergebnis ist eine der Webseite zugeordnete Schlüsselwortliste. Alle gewonnenen Schlüsselwortlisten werden dann zusammengefasst und invertiert. So entsteht ein Index (invertierter Index), der für ein bestimmtes Schlüsselwort auf diejenigen Webseiten verweist, für die dieses Schlüsselwort als relevant eingestuft wurde.
- ❑ Die *Search and Ranking Software* bearbeitet die Suchanfragen und übernimmt eine eventuell sortierte Ausgabe der Suchtreffer. Um eine hohe Relevanz der Suchtreffer zu gewährleisten, kommen hier von Anbieter zu Anbieter unterschiedliche Techniken zum Einsatz. Zu den wohl bekanntesten Ansätzen auf diesem Gebiet zählt der *PageRank-Algorithmus*⁶ von Google, der ein wesentlicher Bestandteil des Erfolgsrezeptes von Google war und ist.

Über die oben genannten Komponenten hinaus, lassen sich noch weitere relevante Subsysteme einer Suchmaschine identifizieren. Diese zu erläutern soll jedoch nicht Bestandteil dieser Arbeit sein. Als weiterführende Literatur, sei hier auf [Glö03] und [Bab01] verwiesen.

2.3 Grenzen heutiger Suchdienste

Die Suchdienste geben zwar eine unverzichtbare Hilfestellung bei der Websuche, sind jedoch mit ihren heutigen Technologien nicht in der Lage die Gesamtheit des WWW zu erfassen bzw. dem Benutzer ausreichend relevante Suchergebnisse zu präsentieren. Trotz

⁵ Synonym für *Webrobot* wird häufig von *Robot*, *Wanderer*, *Crawler* oder auch *Spider* gesprochen. Zum Beschleunigen des „Sammelprozesses“ werden dazu mehrere Webrobots eingesetzt.

⁶ Erklärungen zum Google *PageRank* sind z. B. unter http://www.google.de/intl/de/why_use.html oder in [BP98] zu finden.

ausgeklügelten Suchstrategien ist es den Suchmaschinenbetreibern nicht möglich, die Gesamtheit der im WWW verfügbaren Dokumente in ihre Datenbank aufzunehmen. Vielmehr ergeben Schätzungen über die tatsächliche Menge der im WWW verfügbaren Dokumente im Vergleich mit den Angaben großer Suchmaschinenbetreiber eine Diskrepanz. Es zeigt sich, dass nur ca. 30–40 % aller Dokumente von den Suchmaschinen erfasst werden (Vgl. [Fer03] S. 301 und [GS05]). Nicht nur die ständig wachsende Anzahl der Webdokumente erschweren den Suchmaschinen ihre Arbeit, sondern auch die Art und Weise, wie Informationen im Web abgelegt werden. Babiak identifiziert einige Hauptursachen für Probleme bei der heutigen Informationssuche im WWW: ([Bab01] S. 3 ff.)

- ❑ *Größe*: Die wahre „Größe“ der über das Internet zugänglichen Informationsmenge hat heute ein gigantisches Ausmaß angenommen, so dass des Öfteren sogar von einer Inflation der Information die Rede ist. Die genaue Anzahl der Webdokumente ist unbekannt und kann nur geschätzt werden. So deckt z. B. Google nach eigenen Angaben einen Bestand von über acht Billionen Webseiten ab. Da Information nicht nur aus Webseiten, sondern oft aus Datenbankbeständen die via Web-Schnittstelle abgefragt werden können (das sog. *Deep Web*) zur Verfügung stehen, liegt die wahre Zahl der insgesamt zugänglichen Information um ein vielfaches höher.⁷
- ❑ *Organisation*: Da das Internet keiner zentralen Kontrolle unterworfen ist, existiert auch kein Gesamtkatalog aller Inhalte im Internet.
- ❑ *Strukturierung*: Die Form der im Netz bereitgestellten Informationen reicht von kurzen Texten oder ganzen Seiten mit integrierten Grafiken bis hin zu großen Datenbanken. Diese verschiedenen Arten der Veröffentlichung existieren nebeneinander und bieten keinen einheitlichen Ansatz für eine gezielte Suche.
- ❑ *Dynamik*: Täglich kommen tausende Inhalte hinzu, andere werden gelöscht, verändert oder an eine andere Stelle verschoben.
- ❑ *Qualität*: Der bereits erwähnte inflationäre Charakter der Informationszunahme betrifft neben der Quantität die Qualität der Angebote. Es ist eine Tendenz erkennbar, dass die stetige Zunahme der Webinhalte mit einer gewissen Qualitätsminderung einhergeht. Die große Konkurrenz hat zur Folge, dass viele Webseitenbetreiber lieber eine im Netz bereits vorhandene Informationsquelle neu auf ihren Seiten anbieten, als auf die bereits bestehende Seite (eines vermeintlichen Konkurrenten) zu verweisen. Dadurch werden viele Informationen mehrfach publiziert, wobei die Qualität meist auf der Strecke bleibt.
- ❑ *Homonyme und Synonyme*: Zum einen existieren meist verschiedene Wörter, die ein und dieselbe Bedeutung haben (Synonyme). Zum anderen kann ein einziges Wort mehrere unterschiedliche Bedeutungen aufweisen (Homonyme). Beide Fälle erschweren die Analyse der Inhalte einer Webseite zusätzlich.
- ❑ *Datenformate*: Gemeint ist hier die Einbettung von Information in *Grafiken*, *Java-Applets*, *Flash-Formate* etc. Die Suchmaschine ist in diesem Fall regelrecht „blind“ und kann diese Informationen nicht für eine Auswertung nutzen.

⁷Vgl. hierzu [Ber01]

- ❑ *Manipulationsfähigkeit*: Leider kann den in einer Webseite hinterlegten Daten bei einer Relevanzbewertung nur bedingt vertraut werden. Die Manipulationsversuche der Webseitenprogrammierer zielen meist auf eine „Suchmaschinenoptimierung“ ab, um einen vorderen Platz bei den Suchanfragen zu erhalten. Dies führt mitunter dazu, dass eine Webseite mit populären Schlüsselwörtern angereichert wird, die kaum etwas mit dem wahren Inhalt der Seiten zu tun haben. Die Suchmaschine weist dann solchen Seiten unter Umständen eine vermeintlich positivere Relevanzbewertung bzw. ein höheren Rang zu.

Wie gezeigt wurde, gibt es eine Vielzahl von Schwierigkeiten für eine automatische maschinelle Auswertung. Besonders die fehlende Semantik verhindert weitgehend eine automatische Relevanzbewertung von Webseiten.

Alle bisher besprochenen Techniken, besonders die Ergebnisse der IR-Systeme, erreichen heute nur suboptimale Lösungen. Bei näherer Betrachtung kristallisiert sich eine Barriere heraus, die momentan kaum, und wenn, dann nur mit komplexen und speziellen IR-Ansätzen überwunden werden kann. Die Rede ist von einer der wichtigsten Grundeigenschaften des heutigen WWW: Webseiten werden *von Menschen, für Menschen* bereitgestellt.

Die Aufmerksamkeit liegt darauf, die Information dem Leser optisch adäquat aufbereitet zu präsentieren. Im WWW werden Dokumente dafür zumeist mit Hilfe von HTML so strukturiert, dass ein Browser, welcher die Webseite interpretiert, die Inhalte in eine für den Menschen lesbare Form umwandelt und am Bildschirm darstellt. Eine mögliche maschinelle Auswertung der Dokumente ist auf diese Art nicht möglich.

Webseiten enthalten Meta-Informationen (Informationen über Informationen) darüber, *wie* Inhalte dargestellt werden (z. B. in 14 Punkt großer Arial Schrift). Hingegen ist keine Aussage enthalten, was genau die inhaltliche Bedeutung der Webseite und ihrer Elemente ist und wie die Inhalte eventuell miteinander in Beziehung stehen. Um die Inhalte maschinell sinnvoll weiterverarbeiten zu können, werden zusätzliche Informationen über deren Bedeutung benötigt. Es stellt sich daher die Frage nach einer zusätzlichen Angabe der Semantik. Im nächsten Kapitel wird hierzu eine Erweiterung des WWW vorgestellt, die genau dieses Ziel verfolgt.

2.4 Herausforderung an die Websuche

Bei der Weiterentwicklung von Suchmaschinen sind innovative Ideen gefragt. Eine mögliche Frage dabei lautet: Wie können die heute schon zur Verfügung stehenden Informationen grafisch besser dargestellt werden, so dass mit den vorhandenen Daten eine nutzerfreundlichere Oberfläche erreicht werden kann? Obwohl diese Fragestellung schon lange im Raum steht, gibt es auf diesem Gebiet noch recht wenige Umsetzungen.

Beispielhaft soll hier *Kartoo*⁸ und der Google-Aufsatz *Touch-Graph GoogleBrowser*⁹ aufgeführt werden. Beide gehen bei der Darstellung der Suchergebnisse eigene Wege und stellen die Verlinkungsstruktur zu anderen Webseiten in den Vordergrund.

⁸ <http://www.kartoo.com/>

⁹ <http://www.touchgraph.com/TGGoogleBrowser.html>

Abbildung 1 a) zeigt einen Ausschnitt der Kartoo-Oberfläche. Die Metasuchmaschine Kartoo ordnet die für einen Suchbegriff gefundenen Webseiten nach ihrer Relevanz und Themenzugehörigkeit, um mögliche themengleiche Webressourcen in einer „Wolken“-Darstellung (blaue Bereiche) zu gruppieren. Der Nutzer kann sich interaktiv in der „Hyperlinklandschaft“ bewegen und seine Suche mit einem Klick auf eine Webressource (Knoten) oder einem Link (Kante) verfeinern bzw. steuern. Die auf den ersten Blick recht ungewohnte Suchoberfläche und Bedienung hat nach einer kleinen Eingewöhnungsphase durchaus ihren Reiz. Kartoo bietet dabei einen großen Funktionsumfang, auf den hier nicht weiter eingegangen werden soll.

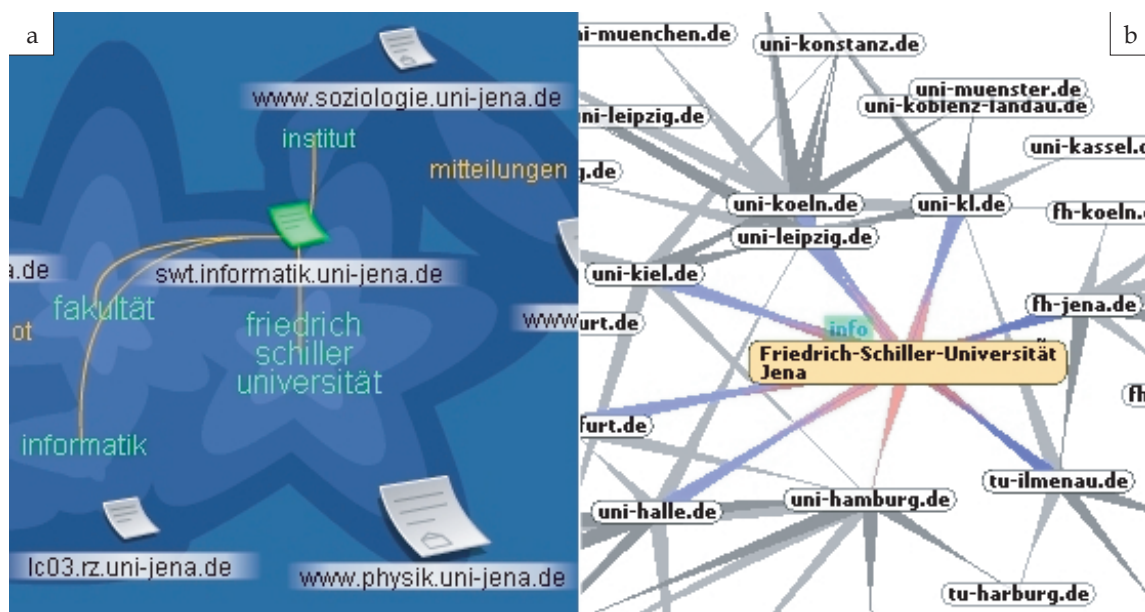


Abbildung 1: Teilausschnitt der Metasuchmaschine Kartoo a) und des TouchGraph GoogleBrowsers b)

Der in Abbildung 1 b) gezeigte Touch-Graph GoogleBrowser stellt für einen eingegebenen URL, die Linkstruktur in Form eines Graphen dar. Die Knoten sind die einzelnen URLs und die Kanten stehen jeweils für die – nach Einschätzung von Google – verwandten Webressourcen. Genutzt wird hierbei ausschließlich die *related-link*-Funktion¹⁰ von Google.

So innovativ und interaktiv die beiden vorgestellten Beispiele auch sein mögen, eine wirkliche prinzipielle Verbesserung der Websuche, bieten diese nach Meinung des Autors nicht. Auf Grundlage der heutigen Situation im Web, bieten obige Beispiele zwar ein Plus an zusätzlicher Information und Benutzerfreundlichkeit, jedoch zeigen sie sich ebenso anfällig in Bezug auf die schon weiter oben besprochenen zahlreichen Probleme bei der Websuche. Kritisch sei hier bemerkt, dass sowohl Kartoo als auch der TouchGraph, sich von der reinen Markupsprachen-Darstellung via HTML entfernt haben. Für Kartoo wird ein Flash-Player-Plugin für den Browser benötigt und für den TouchGraph, über die Javascript Funktionalität hinaus, eine Java Runtime Environment (JRE1.3+), da der Google-Aufsatz ein Java-Applet im Browser ausführt. Diese zusätzlichen technischen Anforderungen an den

¹⁰ Siehe <http://www.google.com/help/features.html#related>

Benutzer-Rechner bzw. -Browser werden die Akzeptanz dieser Suchdienste wohl kaum steigern können.

Obwohl mit Sicherheit das Potenzial – mit Blick auf die visuelle Darbietung – der heutigen Suchmaschinen noch nicht vollends ausgeschöpft ist, scheint die Entwicklung doch auf der Stelle zu treten. Zur Lösung dieses Dilemmas müssen die Voraussetzungen geändert werden. Es muss zusätzlich zu den schon vorhandenen Webseiten, Daten über die Semantik der Inhalte in einer einheitlichen standardisierten Form hinterlegt werden. Der Computer, der heute beim Surfen im WWW zumeist nur noch zu einer bloßen Übertragungs- und Anzeigemaschine degradiert wurde, könnte seine eigentlichen Stärken – das Rechnen und logische Verarbeiten – durch Auswertung dieser Daten vollends ausnutzen. Um dies aber mit Blick auf die Semantik von Informationen erreichen zu können, müssen neue Technologien entwickelt werden, die den Aufbau eines Netzes von semantischen Informationen erlauben. Der nächste Abschnitt beschreibt, wohin die Entwicklung des WWW konkret gehen soll und welche Technologien dafür eingesetzt werden sollen. Obwohl die vorliegende Arbeit nur einen Teil der Idee eines zukünftigen semantischen Netzes aufgreift – nämlich hauptsächlich die Ontologien –, soll das Thema, zur besseren späteren Einordnung dieser Arbeit, im Folgenden vorgestellt werden.

3 Semantic Web

3.1 Die Vision eines semantischen Netzes

„*I have a dream for the Web . . .*“, so beginnt ein Kapitel des von Berners-Lee 1999 veröffentlichten Buches „*Weaving the Web*“ [Ber99]. Nachdem seine Idee des WWW weitgehend realisiert wurde, sieht er seit längerem ein weiteres durchaus mächtigeres Potenzial in einer effektiveren Nutzung von menschlichen Geist und logischer Rechenleistung der Computer. Um dieses Potenzial freizulegen, propagiert er die Schaffung eines semantischen Netzes, dem sog. *Semantic Web*¹¹ (SW). Die Grundidee besteht in der Nutzung der Computer und Netzwerke über ihre bisherige Aufgabe hinaus, so dass eine Kommunikation zwischen Maschine und Maschine über Inhalte einzelner Webressourcen ermöglicht wird. Der heutige Einsatz dieser Systeme auf der Ebene des Internets beruht zum größten Teil auf der Etablierung eines Informationsraumes für die Kommunikation von Menschen über das Medium Internet. Um über diese bloße Bereitstellung eines Informationsraumes hinauszugelangen, müssen als erstes die Daten in einem maschinell auswertbaren (machine-processible) Format vorliegen.

„*The first step is putting data on the Web in a form that machines can naturally understand, or converting it to that form. This creates what I call a Semantic Web – a web of data can be processed directly or indirectly by machines.*“
[Ber99]

¹¹ Der Begriff *Semantic Web*, wurde von Berners-Lee in seiner „Roadmap for a semantic web“ [Ber98] geprägt.

Das World Wide Web Consortium¹² (W3C) beschäftigt sich schon seit den 90er Jahren mit der Frage der Integration semantischer Metadaten in die Struktur des bestehenden WWW. Der SW-Begriff wird auf der SW-Webseite des W3C wie folgt erläutert:

„The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. It is a collaborative effort led by W3C with participation from a large number of researchers and industrial partners. It is based on the Resource Description Framework (RDF), which integrates a variety of applications using XML for syntax and URIs for naming.“ [W3Cb]

Entscheidend ist, dass das SW kein neues Web, sondern eine Erweiterung des heutigen Webs mit wohl definierten Metadaten zur Bedeutungsanreicherung darstellt. Das SW soll damit zusätzlich zu den Daten deren Semantik integrieren. Berners-Lee formuliert die Grundidee kurz und knapp in einem Satz:

„The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.“ [BHL01]

Doch was ist genau unter *wohl definierte Bedeutung* zu verstehen? Am besten macht dies das in Listing 1 gezeigte Gegenbeispiel deutlich:

```
1 <html>
2 <body>
3   <h1>Neu im Programm: Computerdenken</h1>
4   <br>
5   <p>In diesem Buch, zum Preis von nur 39,95 Euro, gibt Roger
6     Penrose einen Einblick in die Debatte um künstliche
7     Intelligenz, Bewusstsein und die Grenzen der Physik.</p>
8   <a href="/bestellung.cgi?id=0815">
9     </a>
10 </body>
11 </html>
```

Listing 1: Quelltext-Beispiel einer HTML-Webseite

Ohne Zweifel enthält die HTML-Seite aus Listing 1 eine Information, die durch Verwendung von HTML-Markup-Tags strukturiert ist: Das Buch „Computerdenken“ ist neu im Programm eines Onlinebuchhändlers. Es werden der Autor, der Preis, und eine kurze Information zum Inhalt des Buches gegeben. Es stellt sich nun die Frage, ob es einen allgemeinen Algorithmus gibt, der aus obiger Repräsentation der Information den Titel, Autor, Preis und Inhalt des Buches zuverlässig extrahieren kann? Die Antwort lautet NEIN. Die Bedeutung ist ohne weitere Hilfe für einen Automaten nicht zu erschließen. Von einer *wohl definierten Bedeutung* kann im obigen HTML-Fragment also keine Rede sein. Ziel des SW ist aber genau dies zu erreichen.

¹² Das World Wide Web Consortium stellt eines der wichtigsten Standardisierungsgremien im Web dar. (<http://w3c.org/>)

Wenn in diesem Zusammenhang von einem maschinellen „Verstehen“ der Daten die Rede ist, so ist damit die Tatsache gemeint, dass Computer eine standardisierte formale Strukturierung der Daten nutzen können, um einzelne Datenteile einer bestimmten Bedeutung zuzuordnen. Ein wirkliches „Verstehen“, wie es sich die Künstliche Intelligenz (KI) wünscht, wird dadurch natürlich nicht realisiert. Die Daten formal mit Semantik zu versehen, ist im Prinzip eine Erweiterung des Metadatenkonzeptes bzw. der Markup-Strategie bei Textannotationen. Hinter dem SW verbirgt sich jedoch mehr als nur eine bloße Metadatenauswertung. Verschiedene *autonome Softwareprogramme*¹³ sollen künftig die auf unterschiedliche Art und Weise kodierte Semantik so verarbeiten können, dass sie den Inhalt eines Webdokumentes sicher einem Bedeutungskreis zuordnen können. Darüber hinaus wäre es einem Agenten möglich, eine inhaltliche Beziehung zu anderen Webdokumenten und deren Semantik herzustellen und schließlich mit Hilfe dieser semantischen Zusatzinformation automatisch und völlig autonom eigene Schlüsse und Ableitungen über die enthaltenen Daten einer Webressource und deren Bedeutung zu ermitteln.

Wenn es gelänge künftig WWW-Dokumente konsequent mit semantischen maschinenlesbaren Metadaten anzureichern, so wäre der Weg frei für eine Nutzung von Computern bei der Gewinnung von Informationen aus dem WWW. Die Webseiten wären dann nicht nur für eine menschliche Auswertung geeignet, sondern zusätzlich ermöglicht ein semantisches Netz ein maschinelles „Verstehen“ und Auswerten der Daten. Ein Computerprogramm wird dadurch in die Lage versetzt, das WWW schnell und effizient völlig selbstständig zu analysieren und könnte den Menschen dabei eine Menge an Arbeit abnehmen. Letztendlich könnte das WWW sein volles Potenzial ausschöpfen und ein Medium mit heute noch ungeahnten Möglichkeiten bieten. Welches Potenzial ein SW entfalten kann, wird anschaulich in dem Artikel [BHL01] anhand eines Einsatzszenarios eines Web-Agents beschrieben.

Die heutige Darbietung der Information im WWW ist nicht dazu in der Lage, eine automatische Auswertung im Hinblick auf die Bedeutung der Daten zu ermöglichen. Zusammenfassend sind die wichtigsten Hindernisse für eine automatische semantische Auswertung nachstehend aufgeführt:

- ❑ Webseiten werden hauptsächlich in HTML verfasst. HTML strukturiert die Information im Hinblick auf die spätere Darstellung, wobei beschrieben wird, *wie* etwas dargestellt werden soll und nicht *was* die Inhalte bedeuten. Einer Maschine ist es in der Regel nicht möglich, damit die Bedeutung der so strukturierten Daten zu erfassen.
- ❑ Das bloße HTML-Konzept der Hyperlinks erlaubt zwar darüber hinaus eine Verknüpfung der Webdokumente untereinander, gibt jedoch auch hier keine Hilfestellung bei der Frage der Semantik.
- ❑ Mit den durch den HTML-Standard bereitgestellten sog. *Meta-Tags* ist prinzipiell eine – wenn auch sehr beschränkte – Bedeutungszuordnung zu einem gesamten Webdokument möglich. Der seltene und oft missbräuchliche Einsatz dieser Tags ist neben der mangelnden Ausdruckstärke der Hauptgrund für die Unzulänglichkeit dieses Ansatzes für die hier diskutierte Problemstellung.

¹³ Es wird hier oft von *Agenten-Systemen* – kurz (engl.) *Agent* – gesprochen.

Wie es gelingen soll, zusätzlich zu dem heutigen bestehenden WWW ein semantisches Netz zu etablieren, wird in den folgenden Abschnitt erläutert.

3.2 Die Semantic Web Architektur

In diesem Abschnitt werden die einzelnen Entwicklungsschritte für eine erfolgreiche SW-Realisierung dargelegt. Abbildung 2 zeigt das momentan favorisierte Architekturmodell des SW, wie es auf den Webseiten des W3C zu finden ist [W3Cb]. Die einzelnen Schichten stellen jeweils eine Abstraktionsebene dar, die alle zusammen genommen das SW bilden.

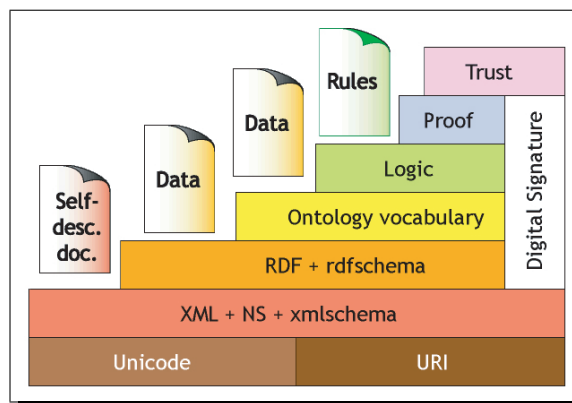


Abbildung 2: Schichtenmodell der Semantic Web Architektur

3.2.1 Unicode + URI

Die Basis aller höheren Ebenen ist zum einen die *Unicode*-Kodierung¹⁴, die die Zeichensätze für fast jede natürliche Sprache bereitstellt und damit die globale Anwendbarkeit des SW garantiert. Zum anderen soll jedes beliebige Objekt über einen *Uniform Resource Identifier (URI)* identifiziert werden können. Der bekannte *Uniform Resource Locator (URL)* stellt einen Teil dieses Ansatzes dar ([MS04] S. 723 ff.). Durch die Verwendung von URIs wird es möglich auch Objekte zu repräsentieren, die nicht durch das WWW abgerufen werden können. Als Beispiel sei hier ein Buch in einer Bibliothek genannt, das auch mit einem URI (hier die ISBN-Nummer) eindeutig identifiziert und referenziert werden kann.

3.2.2 Extensible Markup Language

Für den aus der komplexen *Standard Generalized Markup Language (SGML)* abgeleiteten beschränkten¹⁵ HTML-Sprachstandard hatten sich die Webentwickler damals sehr bewusst – zu Gunsten der Nutzerfreundlichkeit – entschieden. Die relativ leicht erlernbare

¹⁴ Mehr Informationen unter: <http://www.unicode.org>

¹⁵ HTML stellt eine Teilmenge von SGML dar.

Markupsprache bietet einen ausreichend großen Sprachumfang für eine Strukturdefinition von Webdokumenten. Jedoch stellt diese Beschränktheit nun ein Hindernis für die weitere Webentwicklung dar. Wünschenswert wäre eine universelle erweiterbare Markupsprache, welche den Nutzern eine beliebige, nach den eigenen Wünschen und Bedürfnissen angepasste Strukturierung der Daten ermöglicht.

Hierfür kommt XML zum Einsatz, das HTML zukünftig ablösen soll. Bei XML handelt es sich wie bei HTML um eine einfache Teilmenge von SGML, entscheidend ist jedoch die im Gegensatz zu HTML gegebene Möglichkeit der Erweiterbarkeit. Mit XML lässt sich jede gewünschte Syntax realisieren. Der Nutzer kann eigene einfache und komplexere Datentypen frei definieren und den XML-Elementen spezielle Eigenschaften über Attributdefinitionen zuweisen. Die Definition der verwendeten XML-Syntax/Struktur sollte heute vorzugsweise in der *XML-Schema Language (XMLS)* erfolgen.¹⁶ Für mehr Informationen über XML und XML-Schema sei an dieser Stelle auf [ABK+02] verwiesen.

Grammatikdefinitionen ermöglichen die Erstellung einheitlicher XML-Dokumente und die Durchführung von Syntaxprüfungen, allerdings bieten sie keinen Aufschluss über die Bedeutung der einzelnen Elemente. Mit XML können eigene Tags definiert und die Elemente einer Webseite damit gekennzeichnet werden. Dies allein reicht jedoch nicht aus, da z. B. spätere Web-Agenten die Bedeutung der Tags nicht kennen. Das Element ist mit einem bestimmten Namen gekennzeichnet, jedoch kann die Bedeutung ein Computerprogramm nicht ohne weiteres ermitteln. Es herrscht daher nicht nur ein Bedarf an syntaktischer Interoperabilität (wie von XML angeboten), sondern auch an semantischer Interoperabilität. Bei XML ist ein Mangel an deklarativer Semantik festzustellen, da durch XMLS zwar eine Spezifizierung der Syntax ermöglicht wird, jedoch über die genaue Semantik keine einheitliche Vereinbarung besteht. Um die XML-Konstrukte in Beziehung zueinander setzen zu können, muss etwas über deren Bedeutung festgelegt werden können. Einen Mechanismus dafür wird durch das Resource Description Framework bereitgestellt.

3.2.3 Resource Description Framework

Das *Resource Description Framework (RDF)* stellt einen ersten Schritt hin zum SW dar und hat seine Wurzeln in Berners-Lee's SW-Idee. RDF ist seit Februar 1999 eine W3C Recommendation.¹⁷ RDF bildet die Grundlage für die Verarbeitung von Metadaten, anhand derer es möglich wird, einer bestimmten durch einen URI identifizierten Ressource eine Bedeutung zuzuordnen. RDF selbst stellt eine XML-Untermenge dar, deren Vokabular eine fest vorgeschriebene Semantik besitzt. Zu diesem Ansatz existiert beim W3C eine eigene Arbeitsgruppe namens *RDF Core Working Group*¹⁸, die sich mit der Weiterentwicklung und der Etablierung dieses Standards beschäftigt.

Das RDF-Modell bietet prinzipiell eine syntaxunabhängige Darstellungsform für RDF-Ausdrücke und besteht aus drei Objekttypen: [W3Ce]

¹⁶ XML-Schema ist ebenfalls eine beim W3C standardisierte Definitionssprache und soll die nicht auf XML selbst beruhende bisher verwendete *Dokument Type Definition (DTD)* ablösen.

¹⁷ <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>

¹⁸ <http://www.w3.org/2001/sw/RDFCore>

- ❑ **Ressourcen (Resources):** Alle Entitäten in RDF sind Ressourcen. Dies kann z. B. eine gesamte Webseite oder ein Element aus einem HTML- oder XML-Dokument sein. Wichtig ist hierbei, dass Objekte nur dann zu den Ressourcen zählen, wenn sie mit einem URI beschrieben werden können. Es müssen nicht zwangsläufig im Internet erreichbare Objekte sein, eine Zeitschrift oder ein Buch ist ebenfalls denkbar.
- ❑ **Eigenschaften (Properties):** Die Ressourcen sind durch spezielle Eigenschaften definiert und beschrieben. Diese Eigenschaften können durch Randbedingungen genauer spezifiziert werden. Eine Eigenschaft ist somit ein spezieller Aspekt, ein Attribut oder eine Beziehung, die eine Ressource beschreibt oder besitzt.
- ❑ **Beschreibungen oder Aussagen (Statements):** Eine Ressource, in Kombination mit einer namentlichen Eigenschaft und dem Wert für diese Eigenschaft bildet eine Aussage. Dies geschieht, indem man Tripel aus Subjekt-Ressourcen, Prädikaten von Eigenschaften und Werte-Objekte bildet. Ein Objekt kann eine Ressource (URI) oder ein sog. Literal (ein String) sein.

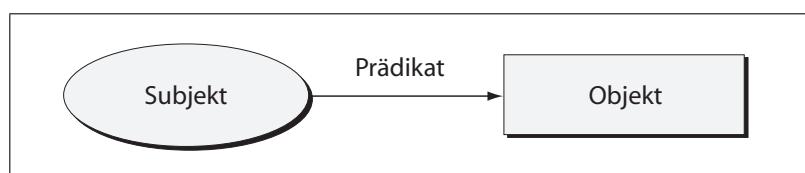


Abbildung 3: Das RDF-Dreigespann

Ein Statement beginnt mit einem Subjekt, gefolgt von einem Prädikat und dem abschließenden Objekt (SPO-Notation). Dieses Dreigespann bildet das RDF-Grundgerüst (siehe Abbildung 3). Es gibt drei unterschiedliche Sichtweisen auf ein Statement.¹⁹ Es sei folgendes Beispiel-Statement gegeben:

Max Mustermann ist Ersteller der Webseite <http://www.max-mustermann.de>.

Die erste Möglichkeit das obige Statement zu interpretieren liegt in der Gruppierung der beteiligten Subjekt-, Prädikat- und Objekt-Ausprägungen:

(„Max Mustermann“, <http://www.beispiel.org/website-creator>, <http://www.max-mustermann.de>)

Es handelt sich um ein Tripel der Form (x, P, y) . Das Prädikat P (hier „ist Ersteller von“) verbindet dabei das Subjekt x („Max Mustermann“) mit dem Objekt y (<http://www.max-mustermann.de>) ausgedrückt durch: $P(x, y)$. Dass es sich hierbei um ein binäres Prädikat handelt ist eine Grundeigenschaft von RDF. (Vgl. [AH04] S. 61 ff.)

Die Darstellung als Graph – auch *N3-Notation* genannt – ist die zweite, für Menschen bevorzugte Darstellungsform. Die Tatsache der binären Prädikate zeigt sich in jeweils einer gerichteten Kante im Graph pro Statement. Es sei hier bemerkt, dass natürlich von einem Subjekt beliebig viele Kanten auf jeweils andere Objekte ausgehen dürfen. Abbildung 4 zeigt die grafische Darstellung des Statements, welches durch die zusätzliche Beschreibung des Titels der Webseite erweitert ist.

¹⁹ Vgl. [AH04] S. 64–66.

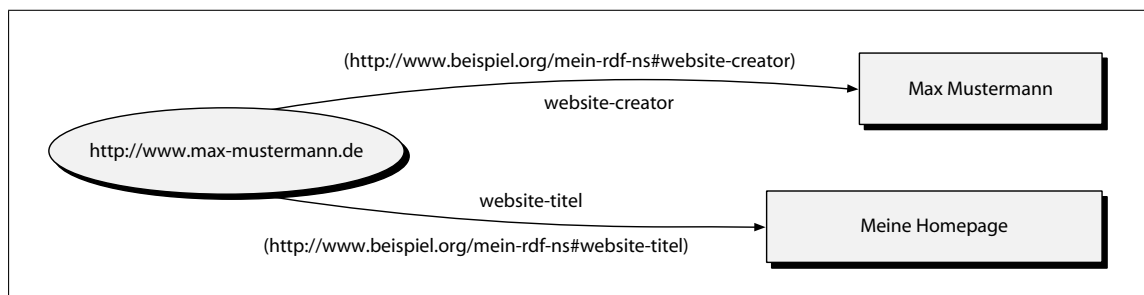


Abbildung 4: RDF-Beziehungen als Graph (N3-Notation)

Das Oval repräsentiert eine Ressource (Subjekt). Die beschrifteten Kanten stellen die Prädikate (Eigenschaften) dar und die Rechtecke²⁰ sind die Objekte (Werte). Es handelt sich hier genauer gesagt um zwei Statements, also zwei Tripel. Das Statement aus Abbildung 4 kann wiederum als eine Ressource in einem weiteren Statement verwendet werden, womit eine Möglichkeit einer beliebigen Schachtelung von Statements (sog. Reifications) gegeben ist.

Die dritte Möglichkeit ein Statement zu repräsentieren, besteht in der formalen Beschreibung mittels XML-Syntax. Da ein Programm diese Repräsentation nur seriell abarbeiten kann, wird diese Repräsentationsform auch als *XML-Serialisation* (RDF-Serialization-Format) bezeichnet. Listing 2 gibt hierfür ein Beispiel.

```

1 <?xml version="1.0" encoding="UTF-16"?>
2 <rdf:RDF
3   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4   xmlns:meine_domain="http://www.beispiel.org/mein-rdf-ns">
5   <rdf:Description rdf:about="http://www.max-mustermann.de">
6     <meine_domain:website-creator>
7       Max Mustermann
8     </meine_domain:website-creator>
9     <meine_domain:website-titel>
10      Meine Homepage
11    </meine_domain:website-titel>
12  </rdf:Description>
13 </rdf:RDF>

```

Listing 2: Beispiel eines Statements in XML-Syntax

Es lassen sich mit RDF sehr komplexe und mächtige Dokumentenstrukturen bilden, auf dessen Einzelheiten hier nicht erschöpfend eingegangen werden kann und soll. Dabei bietet RDF selbst keine Möglichkeit, die Beziehungen zwischen Eigenschaften und Ressourcen zu definieren und gegebenenfalls Restriktionen aufzuerlegen. Um diese Beziehungen und eventuelle Restriktionen zu deklarieren, werden sog. RDF-Schemata (RDFS) eingesetzt. Mit einem RDF-Schema ist es – analog dem XML-Schema – möglich ein eigenes strukturiertes RDF-Vokabular zu definieren. Es bietet definierte Konzepte zur Beschreibung von Klassen, Ressourcen und Eigenschaften sowie deren Zusammenhänge. Mit

²⁰ Alternativ wird hier auch ein Oval verwendet.

RDFS wird eine Typisierung von RDF-Ressourcen ermöglicht. Es können hierarchische Strukturen (Taxonomien) und ihre Beziehungen untereinander modelliert, Datentypen definiert und Restriktionen auferlegt werden. Darüber hinaus sind noch weitere Anwendungen denkbar.²¹

Mit RDF steht eine einfache Modellierungssprache zur Verfügung, die jedoch für eine Formulierung von Wissen allein noch nicht ausreicht. So ist es mit RDF(S) nicht möglich, global definierte Eigenschaften lokal für bestimmte Instanzen einzuschränken. Liegt z. B. eine Eigenschaft *benötigt_Kraftstoff* vor, für die die Klasse *PKW* als *Domain* und die Klasse *Kraftstoff* (mit den Sub-Klassen *Benzin* und *Diesel*) als *Range* definiert ist, ist es nicht möglich einer PKW-Instanz nur den Kraftstoff *Diesel* zuzuordnen. Weiterhin bietet die RDF(S)-Syntax keine Möglichkeit Disjunktheit von Klassen auszudrücken oder neue Klassen durch Kombination aus bereits bestehender zu definieren. Mit RDF(S) können keine Kardinalitätsrestriktionen oder inverse Beziehungen zwischen Klassen formuliert werden.

Für eine größere Freiheit bei der Modellierung semantischer Informationen werden zusätzliche Beschreibungselemente benötigt, die durch den auf RDF/RDFS aufsetzenden Ontology-Layer bereitgestellt werden sollen.

3.2.4 Ontologien (Ontology vocabulary)

Ontologien sind neben XML und RDF die dritte wesentliche Komponente des SW. Im Kontext des SW wird eine Ontologie als eine Kollektion von strukturierten zusammengehörigen Begriffen angesehen, die eine formale Beschreibung eines bestimmten Wissensgebietes (Domäne) bereitstellen. Diese ermöglicht dann eine maschinelle Auswertung der modellierten Wirklichkeit (Konzepte) und erlaubt so die Verbindung von Daten mit Semantik.

Der RDF/RDFS-Layer bietet nur die Möglichkeit Beziehungen zwischen Ressourcen auszudrücken. Ontologien sollen den Ressourcen eine Bedeutung zuordnen, die als eine Art Konvention zwischen Daten und Bedeutung aufgefasst werden kann, welche zuvor von einem Menschen festgelegt werden muss. Im Abschnitt 4 wird das Thema Ontologien ausführlicher erläutert.

3.2.5 Wissensverarbeitung (Logic)

Damit ein Bedeutungsnetzwerk von Maschinen verarbeitet werden kann, muss dieses nicht in der Lage sein z. B. die *wahre* Bedeutung des Wortes „Apache“ genau zu verstehen. Es ist ausreichend ein Regelwerk zu erstellen, mit dem logische Rückschlüsse gezogen werden können (automatic reasoning/inference). Fakten, die durch Ontologien ausgedrückt werden, können als Grundlage für logische Schlussfolgerungen verwendet werden. Ein Schlussfolgerungssystem (Reasoner) kann anschließend z. B. eine Ontologie auf ihre Konsistenz hin überprüfen oder mit Hilfe weiterer referenzierter Ontologien den Wissensvorrat erweitern. (Vgl. [AH04] S. 152 ff.)

²¹ Für nähere Informationen über RDFS siehe [W3Cc].

Ein kurzes und einfaches Beispiel soll dies demonstrieren: Angenommen in einer Ontologie ist festgelegt, dass die Telefonvorwahl von Jena 03641 lautet. In einer weiteren Ontologie über einen Mitarbeiterstamm einer Firma, ist für das Büro des Herrn Mustermanns die Vorwahl 03641 eingetragen. Obwohl die Information über den Ort seines Büros in keiner Ontologie explizit angegeben ist, kann ein Reasoner über die Verbindung der identischen Vorwahlen ermitteln, dass das Büro von Herrn Mustermann in *Jena* liegen muss.

3.2.6 Automatische Beweisführung (Proof)

Für eine aufgestellte Behauptung (Statement) soll eine sog. *Heuristic Engine* solange das SW nach Regeln und Ontologien durchsuchen, bis die Aussage entweder belegt oder widerlegt werden kann. Der Logic-Layer übernimmt das Anwenden und Folgern aus den Regeln. Das automatische Beweisen kann sich dabei als problematisch erweisen, da es unsicher ist, ob ein Beweis überhaupt erbracht werden kann.²² Ein Beweis ist unter anderem dann unmöglich zu erbringen, wenn nicht ausreichend Regeln definiert sind. Ebenso ist es schwer, die erwartete Dauer der Beweisfindung zuvor abzuschätzen. Eine undefinierte, sehr lange Zeitdauer bis zur Lösung ist denkbar, so dass der Prozess scheinbar unendlich viel Zeit benötigt. Ein automatischer Beweis befindet sich demnach bis zu einer eventuellen Lösung in einem undefinierten Zustand. Bis heute gibt es keine praktische Realisierung des Proof-Layers, da hierfür zuerst eine Etablierung der darunter liegenden Schichten notwendig ist.

3.2.7 Vertrauen/Sicherheit (Trust)

Da in einem semantischen Web, wie es bis jetzt besprochen wurde, jeder alles behaupten und definieren kann, ist es notwendig, die vorhandenen Informationen auf ihre Gültigkeit hin zu überprüfen. Wenn eine automatische Folgerung aus und das Beweisen von semantischen Informationen gefordert werden, benötigt man geeignete Vertrauensprinzipien und Authentifizierungsmechanismen. Um die Echtheit einer Information feststellen zu können, soll das Verfahren der *Digitalen Signaturen* verwendet werden. Unter einer Digitalen Signatur versteht man das Verschlüsseln von Daten unter Zuhilfenahme eines sog. öffentlichen (public) und privaten (private) Schlüssels (key) und einem geprüften Echtheitszertifikat. Digitale Signaturen können die Echtheit des Kommunikationspartners bestätigen. Ein manuelles Festlegen glaubhafter Quellen reicht jedoch nicht aus, um alle möglichen Informationsquellen zu beschreiben. Einer Agent-Software muss es möglich sein, ihre eigenen meist implizit im Semantic Web vorhandenen Vertrauensquellen zu kontaktieren und so (z. B. von einer expliziten Vertrauensstelle) eine weitere implizit zu erreichen und deren Informationen mit in die Auswertung einfließen zu lassen. Es muss hier ein Kompromiss zwischen maximaler Vertrauensstellung und realistischer Ergebnisfindung erreicht werden. Die Hoffnung ist, somit ein *Web of Trust* zu schaffen, in dem nur mit wenigen transitiven Schritten eine Vertrauensbeziehung zwischen zwei beliebigen Agenten beschrieben werden kann. (Vgl. [AH04] S. 18)

²² Dieses Problem ist äquivalent zum sog. *Halteproblem* in der Theoretischen Informatik.

4 Wissensbeschreibung durch Ontologien

Menschen greifen bei der Arbeit mit abstrakten Daten jeglicher Art auf persönliches „gespeichertes“ Kontextwissen zurück, welches auf früheren Erfahrungen beruht. Gibt es auf diese Weise keine Lösung, helfen umfangreiche Lehrbücher, Lexika, Fachliteratur oder Regelwerke, einheitliche Konventionen über bestimmte Begriffe eines speziellen Wissensbereiches ausreichend unmissverständlich für Dialoge und Diskurse zu verwenden. Ein Computerprogramm kann im Allgemeinen nicht auf ein derartiges Hintergrundwissen zurückgreifen. Das Konzept der Ontologien soll nun diese Lücke schließen und eine Art formales Nachschlagewerk für Programme bereitstellen.

Der Begriff *Ontologie* ist aus der Philosophie entlehnt und steht dort für die „Lehre vom Sein“ (Vgl. [AH04] S. 10 f.). In weitgehender Anlehnung an diese Bedeutung, bedient sich die Informatik des *Ontologie*-Begriffes. Er bezeichnet hier eine einheitlich zusammengefasste Repräsentation von Begriffen bezogen auf einen zugrunde liegenden speziellen Wissensbereich (knowledge domain). Die Anzahl potenzieller Wissensbereiche ist dabei so groß wie die Vielfalt des kulturellen und wissenschaftlichen Lebens selbst. Wie Hesse in [Hes02] schreibt, macht daher im Kontext der Informatik „... (im Gegensatz zur Philosophie) auch der Gebrauch des Plurals (Ontologien) Sinn“. Eine Ontologie kann definiert werden als eine „... explicit and formal specification of a conceptualization“. [AH04]

Ontologien sind formale semantische Modelle, die dazu dienen, den Austausch und das Teilen von Wissen, insbesondere zwischen menschlichen und maschinellen Akteuren, zu erleichtern. Die einzelnen modellierten Wissensgebiete werden dabei auch Domänen genannt.

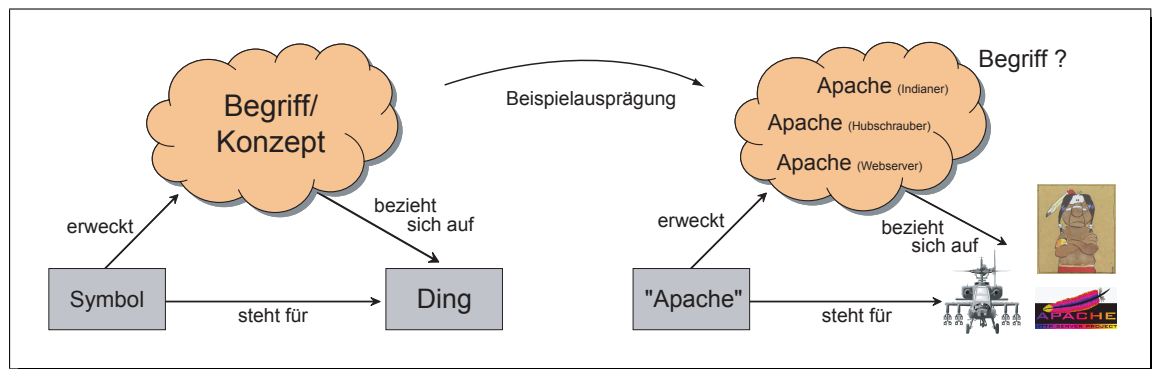


Abbildung 5: Kommunikationssituation – Semiotisches Dreieck

Die Kommunikationssituation, in der sich die Akteure befinden, wird durch das *Semiotische Dreieck* (oder auch *semantisches Dreieck*) beschrieben (siehe Abbildung 5, links). Symbole sind – im Kontext des semiotischen Dreiecks – geschriebene Wörter einer speziellen Sprache. Das Symbol erweckt einen bestimmten Begriff (Konzept), eine kognitive Projektion des Symbols auf einen Gegenstand der realen Welt. Nun sind aber die kognitiven Projektionen aller potenziellen Akteure keineswegs zwingend identisch. Daraus folgt, dass das Symbol selbst, keine eindeutige Assoziationen mit einem Objekt der realen Welt garantiert. Abbildung 5 (rechts) macht anhand eines Beispiels die mehrdeutige Interpretationsmöglichkeit in einer Kommunikationssituation deutlich.

Das Symbol „Apache“ (Abbildung 5 rechts) kann drei mögliche Projektionen auf einen Begriff auslösen. Jeder dieser drei möglichen Begriffe besitzt eigens charakteristische Beschreibungen. Für den Indianer wäre dies z. B. Mensch, amerikanischer Ureinwohner, gehört zum Stamm der Apachen. Alle diese Beschreibungen entsprechen einer speziellen Ontologie oder können zu solchen zusammengefasst werden. Mit ihrer Hilfe soll definiert werden, auf welches Objekt der Wirklichkeit der jeweilige Begriff projiziert werden soll. Mit Ontologien soll Wissen einer Domäne so dargestellt werden, dass innerhalb der Personengruppen eine allgemeine Übereinstimmung, ein Konsens über das Verständnis dieser Domäne gebildet werden kann. Wird die Art der verwendeten Ontologie genau spezifiziert, so ist für jeden Kommunikationsteilnehmer, ob Mensch oder Maschine, eindeutig klar, um welchen Begriff es sich handelt.

Für eine formale Ausgestaltung eines Wissensgebietes werden spezielle Ontologie-Sprachen benötigt, die im Vergleich zu RDF(S) eine differenziertere Beschreibung von Sachverhalten zulassen und so die Ausdruckskraft weiter erhöhen. Dabei sollen Klassen und deren Beziehungen untereinander beschrieben werden, die mit Web-Dokumenten und Anwendungen in Verbindung stehen. Mit Hilfe einer formalen Ontologie-Sprache soll der Mangel an semantischer Ausdruckskraft des RDF(S) Ansatzes ausgeglichen werden.

4.1 Die Web Ontology Language

Die *Web Ontology Language* (OWL)²³ ist eine semantische Auszeichnungssprache (Markup-Sprache) zum Veröffentlichen und Austauschen von Ontologien im WWW. Die Sprache ist eine Weiterentwicklung der Ontologie-Sprache DAML+OIL.²⁴ Die Syntax für OWL ist RDF/XML. OWL ist eine Spezifikation des W3C und hat bei Abschluss dieser Arbeit den Status einer Empfehlung (W3C-Recommendation). Zusätzlich zu RDF und RDF Schema werden weitere Sprachkonstrukte eingeführt, die es erlauben, Ausdrücke ähnlich der Prädikatenlogik zu formulieren. Die OWL-Spezifikation (<http://www.w3.org/2004/OWL/>) besteht aus:

- ❑ **OWL Overview** (<http://www.w3.org/TR/owl-features/>) stellt eine allgemeine Einführung dar.
- ❑ **OWL Guide** (<http://www.w3.org/TR/owl-guide/>) behandelt eine erste Einführung anhand von einigen Beispielen.
- ❑ **OWL Reference** (<http://www.w3.org/TR/owl-ref/>) beinhaltet eine informelle (keine normative!) OWL Referenz.
- ❑ **OWL Semantics and Abstract Syntax** (<http://www.w3.org/TR/owl-semantics/>) stellt die eigentliche und einzige normative Referenz zu OWL dar und ist somit das Hauptdokument der OWL-Spezifikation. (Alle anderen Dokumente haben nur den

²³ Das anfangs verwendete „korrekte“ Akronym WOL missfiel den Entwicklern, woraufhin sie die Sprache in OWL umtaufen. Später wies ein Mitglied des Teams darauf hin, dass in dem bekannten Kinderroman „Winnie the Pooh“ von Milne, die Eule (engl. „OWL“) selbst ihren Namen immer „WOL“ schrieb. In Anlehnung daran war quasi eine nachträgliche Rechtfertigung der Umbenennung gefunden. (Quelle: <http://www.w3.org/2003/08/owlfaq>)

²⁴ Siehe <http://daml.org>

Zweck einer Erklärung, um den Einstieg in die Sprache so weit wie möglich zu erleichtern.)

- ❑ **OWL Test Case** (<http://www.w3.org/TR/owl-test/>) beschäftigt sich mit einigen Testfällen für die Umsetzung der normativen Spezifikation.
- ❑ **OWL Use Case and Requirements** (<http://www.w3.org/TR/webont-req/>) gibt einige Beispiele für Anwendungsfälle von OWL und behandelt allgemeine Anforderungen einer Ontologie-Sprache.

Bei OWL-Dateien handelt es sich zunächst um XML-Dateien. Um die Ausdrucksmächtigkeit von OWL gegenüber XML, XML Schema, RDF und RDF Schema zu erweitern, werden weitere Konstrukte bereitgestellt. Mit OWL können *Klassen* (Classes), *Eigenschaften* (Properties) und *Instanzen* (Individuals) beschrieben werden. Die Klassen stehen dabei für sog. Konzepte, welche spezielle Eigenschaften besitzen können. Instanzen sind hierbei Individuen einer oder mehrerer Klassen.

4.1.1 OWL Sprachebenen

OWL selbst besteht aus einer Menge von drei unterschiedlichen zunehmend komplexeren Sprachversionen: [OWL]

- ❑ **OWL Lite** stellt eine einfache Sprachversion (minimale Untermenge von OWL) zur Umsetzung einfacher Klassifikationshierarchie und einfachen Beschränkungseigenschaften dar. „OWL-Lite“ soll für die Entwickler einen Kompromiss zwischen Nützlichkeit und Zugänglichkeit darstellen, um die Akzeptanz zu forcieren.
- ❑ **OWL DL** beinhaltet den vollständigen OWL-Wortschatz, welcher unter einer Anzahl einfacher Beschränkungen interpretiert wird. DL steht hier für *Description Logic* (Begriffslogik), die semantische Netze in ihrer Ausdruckstärke erweitert. Dabei ist OWL DL am ehesten mit DAML+OIL vergleichbar.
- ❑ **OWL Full** umfasst den gesamten OWL-Wortschatz plus der vollständigen RDF-Syntax. Damit ist eine Erstellung von Ontologien in einer reinen RDF-Syntax möglich. OWL Full-Dokumente sind daher zugleich RDF-Dokumente (und umgekehrt! [HL04]) und bieten die mächtigste Möglichkeit zur Modellierung einer Ontologie – jedoch bei einer nach oben offenen Komplexitäten.

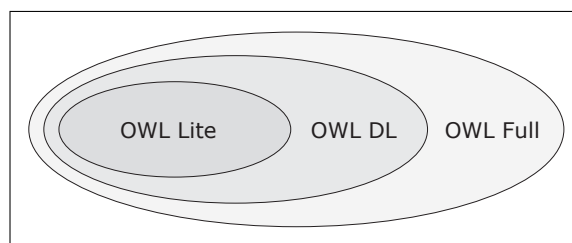


Abbildung 6: OWL Sprachebenen

4.1.2 Aufbau einer OWL-Datei

Eine detaillierte Darbietung des vollen Sprachumfanges ist im Rahmen dieser Arbeit nicht möglich. Vielmehr wird im Folgenden der prinzipielle Aufbau einer OWL-Datei und wichtiger Sprachelemente besprochen. Für eine volle Einarbeitung in das Thema ist ein Studium der umfangreichen OWL-Spezifikation [OWL] notwendig.

Beginnend mit der für XML-Dateien notwendigen `<?xml version="1.0"?>` Angabe, folgt in einem öffnenden `rdf:RDF`-Tag die Deklaration der XML-Namensräume des verwendeten Vokabulars (siehe Listing 3).

```

1 <?xml version="1.0"?>
2 <rdf:RDF
3   xmlns="http://www.artusweb.de/SontoX/ontology/fsu-jena.owl#"
4   xml:base="http://www.artusweb.de/SontoX/ontology/fsu-jena.owl"
5   xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
6   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
7   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
8   xmlns:owl="http://www.w3.org/2002/07/owl#">
9   (...)

```

Listing 3: XML-Namensraum Deklaration

Der Ontologie-Kopf (OWL-Header), eingeschlossen in dem `owl:Ontology`-Element, bietet die Möglichkeit Angaben über das OWL-Dokument selbst zu formulieren. Möglich sind hier z. B. eine Benennung der Ontologie (`rdfs:label`), ein Kommentar (`rdfs:comment`), die Angabe von Versionsinformationen (`owl:versionInfo`, `owl:priorVersion`) oder das Importieren einer anderen Ontologie (`owl:import`). Listing 4 zeigt einen beispielhaften OWL-Header:

```

1 (...)
2 <owl:Ontology rdf:about="">
3   <owl:versionInfo>1.0 2005/06/04</owl:versionInfo>
4   <rdfs:label xml:lang="de">Beispiel-Ontologie</rdfs:label>
5   <owl:imports rdf:resource="http://protege.stanford.edu/plugins/owl/
6     /protege"/>
7   <rdfs:comment xml:lang="de">Beschreibung der Ontologie...
8   </rdfs:comment>
9 </owl:Ontology>
10 (...)

```

Listing 4: OWL-Header Definition

Zwischen Header und dem schließenden `</rdf:RDF>`-Tag erfolgt die eigentliche Definition der Wissensdomäne. Als erstes wird ein Satz von Wurzelklassen²⁵ der Domäne mit Hilfe von OWL-Basisdefinitionen definiert (`owl:Class`). Darauf aufbauend können speziellere Klassen abgeleitet werden (`rdfs:subClassOf`)²⁶, wodurch eine Taxonomie der Klassenstruktur ausgedrückt wird. Listing 5 zeigt hierfür ein einfaches Beispiel:

²⁵ Das eigentliche Wurzelement, von dem alle OWL-Elemente abgeleitet sind, ist das Element `owl:Thing`. Jede nutzerdefinierte Klasse ist daher implizit eine Unterklasse von `owl:Thing`.

²⁶ Das `subClassOf`-Element besitzt dabei die Eigenschaft der Transitivität.

```

1  (...)
2  <owl:Class rdf:ID="Person"/>
3  <owl:Class rdf:ID="Abteilung"/>
4  <owl:Class rdf:ID="Geschlecht"/>
5  <owl:Class rdf:ID="Frau">
6    <rdfs:subClassOf rdf:resource="#Person"/>
7    <owl:Restriction>
8      <owl:onProperty rdf:resource="#geschlecht"/>
9      <owl:hasValue rdf:resource="#weiblich" rdf:type="#Geschlecht"/>
10   </owl:Restriction>
11 </owl:Class>
12 (...)

```

Listing 5: OWL-Class Definition

Listing 5 zeigt des weiteren in Zeilen 7–10 ein Beispiel für eine Eigenschaftsbeschränkung (*owl:Restriction*), die ausdrückt, dass die Klasse *Frau*, für die Eigenschaft des *Geschlechtes* den Wert *weiblich* besitzt. Über das Element *Restriction* lassen sich weiterhin Beschränkungen der Kardinalität (*owl:cardinality*) definieren. Mit *owl:maxCardinality* und *owl:minCardinality* (min:max-Notation) ist es möglich einen Wertebereich festzulegen.

Über eine bloße Klassen-Taxonomie hinaus, lassen sich mit der Definition von Eigenschaften (Properties) Aussagen über Klassen und davon abgeleiteten Elementen (Individuen) treffen (Listing 6). Zu den beiden wichtigsten Typen von OWL-Properties gehören:

- ❑ **ObjectProperty:** Mit Hilfe einer Objekteigenschaft (*owl:ObjectProperty*) werden Beziehungen zwischen Elementen von Klassen definiert.
- ❑ **DatatypeProperty:** Die Datatypeeigenschaft (*owl:DatatypeProperty*) erlaubt die Zuordnung eines XML-Datentypes zu einem Element.

```

1  (...)
2  <owl:ObjectProperty rdf:ID="geschlecht"
3    rdf:type="http://www.w3.org/2002/07/owl#FunctionalProperty">
4    <rdfs:range rdf:resource="#Geschlecht"/>
5    <rdfs:domain rdf:resource="#Person"/>
6  </owl:ObjectProperty>
7  <owl:DatatypeProperty rdf:ID="Abteilung">
8    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
9    <rdfs:domain rdf:resource="#Abteilung"/>
10 </owl:DatatypeProperty>
11 (...)

```

Listing 6: OWL-Property Definition

Nach der Klassendefinition ist es nun möglich davon abgeleitete Elemente (Individuen) zu definieren. Der einfachste Weg dies zu tun zeigt Listing 7.

```
1  (...)
2  <Person rdf:ID="Max Mustermann">
3  <Person rdf:ID="Susi Sorglos">
4      <geschlecht rdf:resource="#weiblich"/>
5      <Abteilung rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
6          Rechnungswesen</Abteilung>
7  </Person>
8  (...)

```

Listing 7: OWL-Individual Definition

Obige Definitionen stellen das Grundgerüst eines OWL-Dokumentes dar. OWL stellt darüber hinaus eine Vielzahl von komplexen und erweiterten Modellierungsmöglichkeiten bereit, die hier im Detail nicht alle behandelt werden können. Stellvertretend seien einige der erweiterten OWL-Sprachmittel genannt:

- ❑ Durch die Definition sog. Eigenschaftsmerkmale werden erweiterte Schlussfolgerungen ermöglicht. So können transitive (*TransitiveProperty*), symmetrische (*SymmetricProperty*) und funktionale (*FunctionalProperty*) Eigenschaften festgelegt werden. Mit *inverseOf* können inverse Eigenschaften und mit *InverseFunctionalProperty* inverse funktionale Eigenschaften definiert werden.
- ❑ Zusätzlich zu den schon oben besprochenen Beschränkungen von Eigenschaften durch *cardinality* und *hasValue* stehen die Konstrukte *allValuesFrom* und *someValuesFrom* zur Verfügung.
- ❑ Mit *equivalentClass* und *equivalentProperty* ist es möglich auszudrücken, dass einzelne Klassen oder Eigenschaften aus einer Ontologie äquivalent zu Klassen und Eigenschaften einer anderen Ontologie sind.
- ❑ Durch *sameIndividualAs* wird (ähnlich wie bei den Klassen) ausgedrückt, dass zwei verschiedene Individuen in ihrer Bedeutung identisch sind. Das Konstrukt *differentFrom* drückt genau das Gegenteil aus.
- ❑ OWL-Klassenerweiterungen stellen Mengen dar, auf denen Mechanismen für die Bildung von Durchschnitt (*intersectionOf*), Vereinigung (*unionOf*) und Komplement (*complementOf*) existieren.
- ❑ Eine Separierung von Mengen von Klassen kann mit *disjointWith* ausgedrückt werden. Hiermit wird sichergestellt, dass ein Individuum als Element einer Klasse, nicht gleichzeitig eines einer anderen Klasse sein darf.
- ❑ Einzelnen Klassendefinitionen können mit *oneOf* zu einer einzigen aufzählenden Definition zusammengefasst werden.

Alle aufgezählten OWL-Sprachkonstrukten sind in ihrer Verwendbarkeit abhängig von der verwendeten OWL-Sprachebene und stellen einen Teil des vollen OWL-Sprachumfangs dar. Eine komplette Auflistung aller Sprachelemente und Hinweise zu ihrer korrekten Verwendung findet sich auf den Spezifikations-Webseiten: <http://www.w3.org/2004/OWL/>.

4.2 Ontologie-Editoren

Eine Schlüsselrolle für die erfolgreiche Entstehung eines SW nimmt die Entwicklung spezieller Software-Tools für die Erstellung von Ontologien ein. Eine mögliche weite Verbreitung von Ontologien wird nur dann Erfolg haben, wenn dem Nutzer ausreichend ausgereifte Tools zur Erstellung und Verwaltung eigener Ontologien zur Verfügung stehen. Obwohl sich die Idee des SWs wachsender Beliebtheit erfreut, existieren bisher nur wenige Applikationen für eine Ontologieentwicklung. Speziell für die junge Ontologiesprache OWL stehen dem Entwickler momentan wenige Tools zur Verfügung. Hinsichtlich der Entwicklungsumgebungen nimmt das *Protégé*-Projekt der Universität Stanford eine führende Stellung ein. Die Editoren *OilEd* von der University Manchester und *OntoEdit* von der Firma OntoPrise sind zwei weitere umfangreiche Vertreter für Ontologieeditoren, welche zur Vollständigkeit erwähnt werden.²⁷

4.2.1 Das Protégé-Projekt

Bei *Protégé*²⁸ handelt es sich um ein Open-Source-Projekt der Universität Stanford, welches seit Frühjahr 2005 als Version 3.0 erhältlich ist. Das mit einem *Graphical User Interface (GUI)* ausgestattete mächtige Entwicklungstool (Abb. 7) zur Ontologiemodellierung bietet über einen Ontologieeditor hinaus zusätzlich einen Knowledge-Base-Editor. *Protégé* ist in Java geschrieben und bietet durch den auf der Jena2-API²⁹ basierten OWL-Plugin eine OWL-Unterstützung. Dadurch ist es möglich, eine auf OWL basierende Ontologie zu erstellen und zu editieren. Weiterhin ist eine Ontologie-Validierung schon während der Erstellung selbst möglich, wodurch Modellierungsfehler bereits zu Beginn erkannt und damit vermieden werden können.

Der anfangs noch recht unhandlich wirkende Editor, erweist sich nach kurzer Einarbeitungszeit schnell als ein unentbehrliches Hilfsmittel. Die Benutzeroberfläche bietet eine Anzahl von Registerkarten mit deren Hilfe die Erstellung bzw. das Editieren von Ontologien ermöglicht wird. Hierbei sind die wichtigsten die Registerkarten *OWL-Classes*, *Properties*, *Individuals* und *Forms*. Für den Aufbau einer Wissensbasis werden zuerst über *OWL-Classes* die verschiedenen benötigten Klassen definiert. Über die *Properties* können die Eigenschaften, die eine Klasse bzw. eine spätere Instanz der Klasse haben soll, festgelegt werden. Da die Klassendefinition zuvor schon umgesetzt wurde, kann nun komfortabel zur Definition des Werte- (Range) und Definitionsbereichs (Domain) der Property auf die Klassen zugegriffen werden. Auf der Reiterkarte *Individuals* können im Anschluss beliebig viele Instanzen einer Klasse angelegt und die durch die Properties zuvor bestimmten möglichen Attributwerte ausgefüllt werden. Es wird an dieser Stelle auch von *Wissensakquise* gesprochen, da das Anlegen von Instanzen eine konkrete Realisierung eines Wissensbereiches, anhand einer zuvor definierten („leeren“) Ontologie entspricht. Hinter

²⁷ Es sei hier auf die Kurzstudie [Tie03] verwiesen. Der Leser erhält dort auf knapp 20 Seiten einen ersten groben Überblick über existierende Software-Tools zum Ontologiemanagement.

²⁸ Homepage des Protégé Projekts: <http://www.protege.stanford.edu>

²⁹ Jena2 ist eine Open-Source Java Framework API speziell für SW-Applikationen und stammt aus den Labors von Hewlett Packard. (<http://jena.sourceforge.net>)

dem *Forms*-Konzept verbirgt sich die Absicht, die Wissensakquise über spezielle Formulare zu steuern und zu erleichtern. Dem Entwickler steht es frei, die verschiedenen Teile der Eingabeformulare nach eigenen Vorstellungen auf der Oberfläche zu arrangieren, Teile zu verbergen oder bestimmte Feldrestriktionen zu definieren. Der Entwickler kann damit schon im Vorfeld Einfluss darauf nehmen, wie genau die eventuelle spätere Wissensakquise zu erfolgen hat.

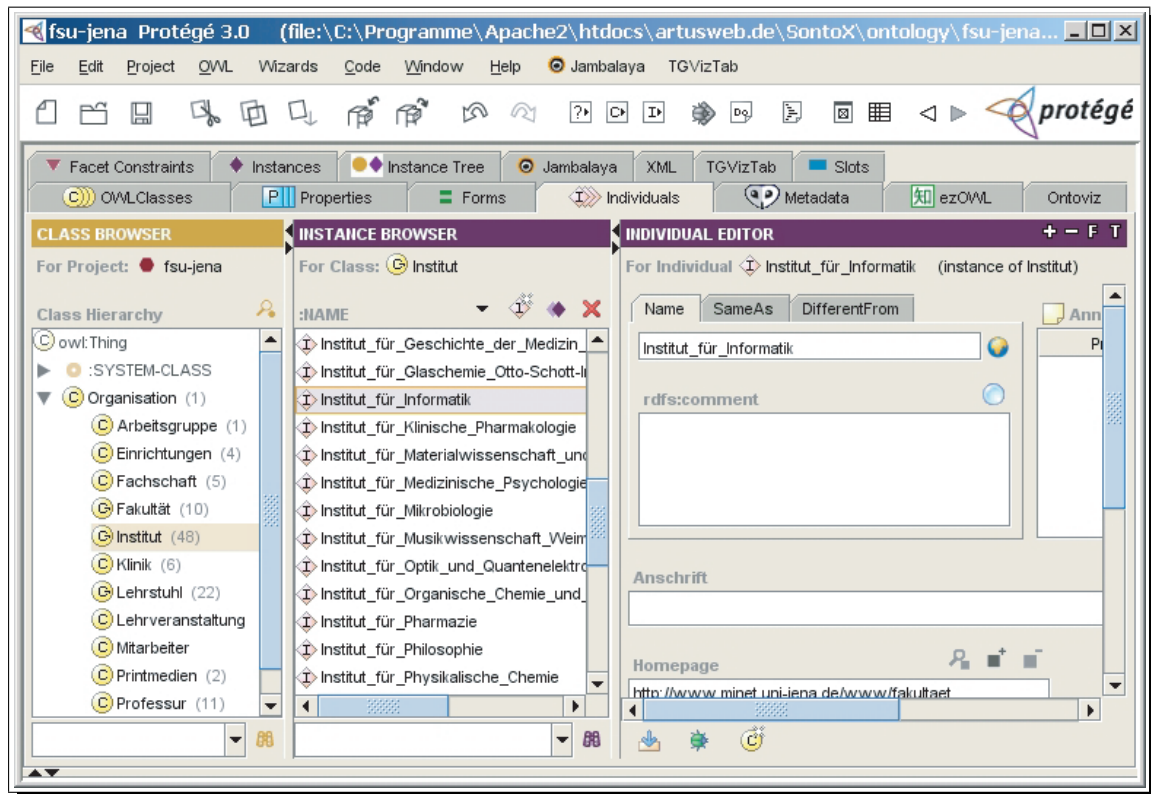


Abbildung 7: Das GUI von Protégé V3.0

4.2.2 Plugins für Protégé

Zu Protégé existieren eine Vielzahl von Plugins, die über die Grundfunktionalitäten hinaus weitere Hilfestellungen für eine Ontologiemodellierung bzw. -verwaltung geben können. So bieten z. B. die Plugins *OntoViz* und *Jambalaya* eine grafische Visualisierung, die besonders bei komplexeren Ontologien dem Entwickler einen erleichterten Überblick verschafft. Auf den Projektseiten von Protégé ist ein Überblick über alle aktuellen zur Verfügung stehenden Plugins³⁰ zu finden.

Stellvertretend für die vielen Plugins, zeigt Abbildung 8 einen Auszug aus dem *Jambalaya*-Plugin³¹. Zu sehen ist eine „radiale“ Visualisierung eines Auszuges einer Ontologie, wobei ein hellblaues Quadrat jeweils ein Individuum einer bestimmten Klasse (gelbe Quadrate) darstellt. Die Dreiecke stehen für zusammengefasste, in der Hierarchie tiefer liegende Instanzen oder Klassen. Das Plugin ermöglicht darüber hinaus noch eine Vielzahl von

³⁰ <http://www-protege.stanford.edu/plugins>

³¹ Projekthomepage von *Jambalaya*: <http://www.thechiselgroup.org/jambalaya>

modifizierten Darstellungsformen, die interaktiv bei der Navigation „durch“ die Ontologie genutzt werden können.

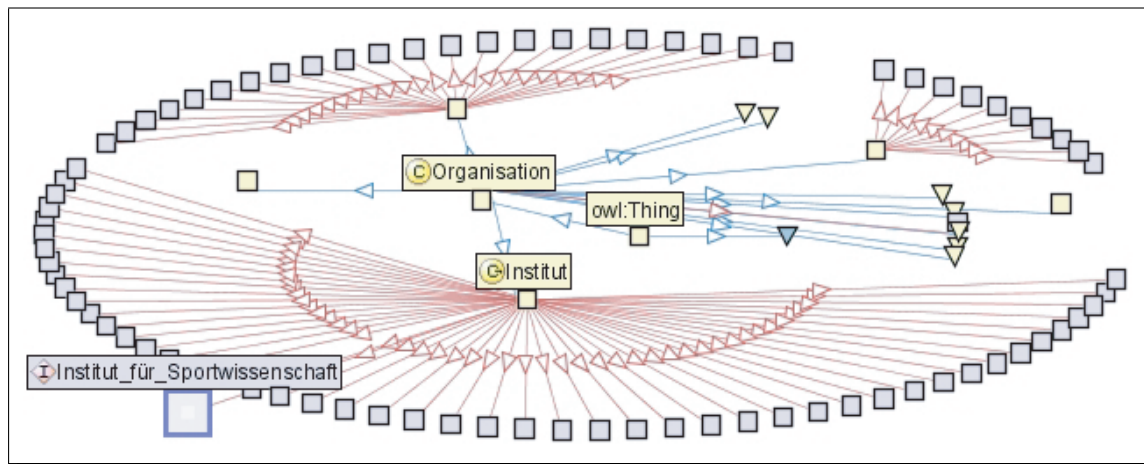


Abbildung 8: Hierarchiedarstellung mit dem Plugin Jambalaya.

Abschließend bleibt zu bemerken, dass jeder, der eine Ontologie modellieren, verwalten oder erweitern möchte, kaum auf die Hilfe von Protégé verzichten kann. Die Bereitstellung dieses mächtigen Tools, eröffnet auch denjenigen unter den Nutzern, die in der Sprachsyntax von Ontologiesprachen weniger bewandert sind, eine einfache Möglichkeit, in das Thema Ontologie einzusteigen und es für sich zu gewinnen. Dies würde einen wichtigen Teil der Vision des SWs voranbringen, nämlich die schrittweise Akzeptanz von Ontologien und somit auch ihre zwingend notwendige Verbreitung. Mit einer stetig wachsenden Zahl an Ontologien im Netz, erscheint auch eine relevante Nutzung dieser Wissensbasen für kommende Semantic Web Anwendungen immer größer, was wiederum das Bereitstellen weiterer Ontologien zur Folge hat, usw.

5 Web Services

Ein Teil der SW-Vision besteht in der Möglichkeit der teils autonom ablaufenden Kommunikation zwischen Maschinen. Dies ist heute kein Wunschtraum mehr, sondern mit dem sog. *Web Services*-Ansatz schon seit einigen Jahren umgesetzt. *Web Services* sind spezielle Dienste, welche eine Kommunikation zwischen Maschinen erlauben. Der Dienstanbieter (der Server) stellt einen gewissen Dienst für andere Rechner (die Clients) zur Verfügung. Die Idee basiert auf dem Server-Client-Prinzip, wobei der Client eine speziell formulierte Anfrage an den Server stellt, welcher in Abhängigkeit seines angebotenen Dienstes die Anfrage bearbeitet und das Ergebnis an den Client zurückgibt. Die *Web Service Architecture Working Group*³² des W3C definiert Web Services wie folgt:

„A Web service is a software application identified by a URI, whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via internet-based protocols.“

³² <http://www.w3.org/TR/2002/WD-wsa-reqs-20021011#IDAGWEBD>

Obwohl diese Technologie noch in den Kinderschuhen steckt, und es heute nur eine geringe Zahl von Implementierungen im WWW gibt, haben die Web Services ihre Feuertaufe überstanden und sind auf dem besten Weg eine große Rolle in der zukünftigen Servicelandschaft des WWW zu spielen. Die Schlüsselfrage bei dieser Technologie besteht in der Bereitstellung eines geeigneten Protokolls, auf dessen Basis ein standardisierter Kommunikationsfluss ablaufen kann. In diesem Bereich hat sich das sog. *SOAP*-Protokoll etabliert. Im Zusammenhang mit diesem Ansatz wird auch von einer Middleware-Architektur gesprochen. Abbildung 9 zeigt die service-orientierte Web Services-Architektur, wobei drei auf XML basierende Standards die Grundlage bilden:

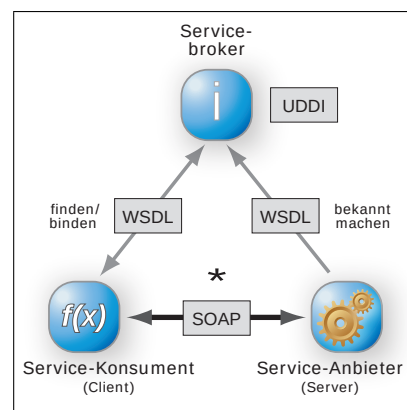


Abbildung 9: Schema der service-orientierten Web Service-Architektur

- ① **SOAP:** Der Name SOAP³³ steht für einen Kommunikationsprotokoll-Standard, der die Kommunikation von im Internet verteilten Applikationen unabhängig von der zugrunde liegenden Software-Architektur, über XML-Nachrichtenaustausch regelt. Im nächsten Abschnitt werden die wichtigsten Merkmale von SOAP kurz umschrieben.
- ② **Web Service Description Language (WSDL):** Hierbei handelt es sich um einen Standard zur Dienst-Beschreibung von Web Services. Eine WSDL-Beschreibung wird mit Hilfe von XML formuliert und beinhaltet unter anderen Informationen über Datentypen, Methoden und Daten einer Nachricht und die möglichen Übertragungsprotokolle. (Siehe [WSDL, HL04].)
- ③ **Universal Description, Discovery and Integration (UDDI):** UDDI ist ein auf XML basierender Verzeichnisdienst-Standard, der es einem Service-Anbieter ermöglicht, seinen Web Service zu publizieren. Ein Nutzer kann daraufhin in einem UDDI-Verzeichnis³⁴ nach einem bestimmten Web Service suchen. UDDI ermöglicht eine Registrierung, eine spezifische Suche und eine Schnittstellendefinition. Der Standard befindet sich in der Obhut des OASIS-Konsortiums (Organisation for the Advancement of Structured Information Standards) und ist aktuell in der Version 3 verabschiedet. [UDDI]

Eine zusammenfassende Informationsquelle zum Begriff Web Service findet sich unter [Jec04]. Darüber hinaus wird in [DJ04] die Stellung der Web Service-Technologie im Kontext des SW diskutiert. Im nächsten Abschnitt wird das SOAP-Kommunikationsprotokoll näher betrachtet.

³³ SOAP stand zu Beginn für Simple Object Access Protocol, da es jedoch weder simpel noch direkt zum Objektzugriff dient, entschied die zuständige W3C Arbeitsgruppe, dass SOAP ab der Version 1.2 kein Akronym mehr ist, sondern nur noch für sich selbst steht. [HL04]

³⁴ Im Web existieren dafür verschiedene Anlaufstellen. Z. B.: <http://uddi.microsoft.com> (Microsoft), <http://uddi.ibm.com/ubr/registry.html> (IBM) oder <http://uddi.sap.com> (SAP).

5.1 Das Web Service-Protokoll SOAP

Bei SOAP handelt es sich um ein aus XML-RPC³⁵ abgeleitetes und auf XML basierendes Protokoll zur Nachrichten-Übermittlung. Das von IBM weiterentwickelte Protokoll, welches speziell die Kommunikation zwischen Maschine und Maschine regelt, stellt einen wichtigen Bestandteil der Web Service-Architektur dar. SOAP wurde von IBM im April 2000 beim W3C als Vorschlag eingereicht und ist momentan in der Version SOAP 1.2 standardisiert. Das Protokoll verwendet zum Nachrichtenaustausch zwischen den Kommunikationspartnern das XML-Format.

Nach [HL04] sind für ein Konzept eines Protokolls zur Nachrichten-Übermittlung drei Fragen von Bedeutung:

- ① *Wie wird eine Nachricht genau übermittelt?*
- ② *Wie ist die Nachricht aufgebaut?*
- ③ *Wie genau sieht der Inhalt der Nachricht aus?*

5.1.1 Übermittlung der Nachricht

SOAP-Nachrichten können entweder als einfache Nachrichten in eine Richtung versandt werden oder – wie bei XML-RPC – in Form einer Anfrage und Antwort ablaufen, wobei die Antwort einen Methodenaufruf mit Rückgabewert darstellt. Der Pfad, den die Nachricht vom Sender zum Empfänger nimmt, wird „Message Path“ genannt und führt über drei unterschiedliche Arten von Knoten (Nodes): [SOAP]

- ❑ **SOAP sender:** Der Sender der SOAP-Nachricht, der diese zuerst abschickt.
- ❑ **SOAP intermediaries:** Intermediäre sind Zwischenstationen, die die Nachricht empfangen und weiterleiten können.
- ❑ **SOAP receiver:** Der Empfänger, der die Nachricht endgültig erhält und verarbeitet.

Damit sich Sender und Empfänger finden, enthält die Nachricht den URI des Empfängers – plus einen eventuellen zusätzlichen URI für den nächsten Intermedian. Der Versand einer SOAP-Nachricht wird zumeist über das Transportprotokoll HTTP realisiert. Da HTTP ein zustandsloses Transportprotokoll³⁶ ist, ist auch jede SOAP-Nachricht zustandslos, d. h. auf dem Weg ist der aktuelle Status der Nachricht nicht bekannt.

³⁵ XML-RPC (XML- Remote Procedure Calls) ermöglicht Methoden- oder Funktionsaufrufe über ein Netzwerk. Entwickelt wurde XML-RPC 1998 von Dave Winer, der die Idee hatte XML und HTTP so zu verbinden, dass ein Nachrichtenaustausch via XML über das Internet möglich ist. Weiterführende Informationen unter: <http://www.xmlrpc.com>

³⁶ Siehe [MS04] S. 735 ff.

5.1.2 Aufbau der Nachricht

Eine SOAP-Nachricht kann in drei Teile untergliedert werden: Wie eine Art Umschlag fungiert der *SOAP-Envelope*, den Datenkopf der Nachricht stellt der *SOAP-Header* dar und der *SOAP-Body* beinhaltet den eigentlichen Nachrichtenteil. Wie Abbildung 10 zeigt, wird der Header und der Body in dem Envelope gekapselt. Dieses Paket wird dann wiederum in das verwendete Transportprotokoll (hier HTTP) eingebettet. Listing 8 zeigt in Anlehnung an [HL04] ein Beispiel einer per HTTP-Request übermittelten SOAP-Nachricht, wobei die Angabe des Headers hier nur zur Demonstration hinzugefügt wurde. Der SOAP-Header ist optional und fehlt bei vielen Implementierungen ganz.³⁷

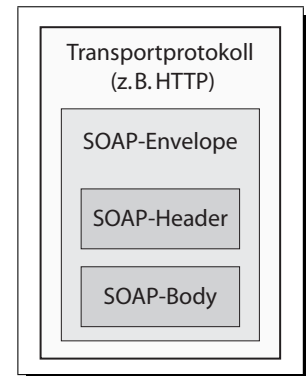


Abbildung 10: Aufbau einer SOAP-Nachricht.

```

1 POST /Ausgabe/ausgabe.asmx HTTP/1.1
2 Host: localhost
3 Content-Type: text/xml; charset=utf-8
4 Content-Length: length
5 SOAPAction: "http://www.ein-beispiel.de/hallowelt/ausgeben"
6 <?xml version="1.1" encoding="utf-8"?>
7 <SOAP:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope
  /"
8         xmlns:xsd="http://www.w3.org/2001/XMLSchema"
9         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
10   <SOAP-ENV:Header>
11     <un:Username xmlns:un="http://www.ein-beispiel.de/username"
12       SOAP-ENV:mustUnderstand="1"> Max Mustermann
13     </un:Username>
14   </SOAP-ENV:Header>
15   <soap:Body>
16     <ausgeben xmlns="http://www.ein-beispiel.de/hallowelt">
17       <text>Hallo Welt</text>
18     </ausgeben>
19   </soap:Body>
20 </soap:Envelope>
  
```

Listing 8: Beispiel einer einfachen SOAP-Nachricht.

Im Folgenden soll das einfache Beispiel aus Listing 8 kurz genauer betrachtet werden:

Zeilen 1–5 stellen den HTTP-Header dar. Da die Nachricht mit HTTP übertragen wurde, steht der HTTP-Header an erster Stelle.³⁸ In Zeile 6 folgt der erste Bestandteil der SOAP-Nachricht. Da diese auf XML basiert, wird an dieser Stelle das für XML-Dokumente typische „Einleitungs“-Tag mit der XML-Version und dem Zeichensatz angegeben. Die Zeilen 7–20 geben den SOAP-Envelope (plus eingebetteten Header und Body) an, welcher zwingend für eine SOAP-Nachricht erforderlich ist. Innerhalb des *Envelope*-Tags folgt die Deklaration der benötigten Namespaces, wie z. B. für das XML-Schema (Zeile 8, 9) und der

³⁷ Siehe [HL04] S. 52 f.

³⁸ Siehe [MS04] S. 745 ff. oder in RFC 2516 (Hypertext Transfer Protocol – HTTP/1.1).

SOAP-Namespace (Zeile 7) selbst. Im SOAP-Header (Zeilen 10–14) stehen meist Informationen zur Autorisierung und Authentifizierung des Senders und/oder spezielle Steuerungshinweise für die Nachrichtenübermittlung. Der Wert 1 (entspricht *true*) für das *mustUnderstand*-Attribut weist dem Empfänger der Nachricht an, dass er den Header kennen muss. Ein weiteres Beispiel für eine Transaktionssteuerung ist z. B. das *relay*-Attribut. Hat es den Wert 1 (*true*), wird der Intermediär angewiesen das entsprechende Header-Element weiterzuleiten.³⁹ In Zeilen 15–19 steht die eigentliche SOAP-Nachricht eingefasst in dem *soap:Body*-Tag. Ein SOAP-Body muss, im Gegensatz zum SOAP-Header, in jeder Nachricht enthalten sein.

5.1.3 Inhalt der Nachricht

Damit eine Kommunikation zwischen Anbieter und Konsument Erfolg hat, muss eine SOAP-Nachricht exakt und unmissverständlich nach den Spezifikationen des Web Services verschlüsselt werden. Die Verschlüsselung (*Encoding*) übernimmt die Aufgabe, die Methodenaufrufe und deren Parameter (plus Datentypen) in XML zu verschlüsseln (XML-Serialisation). Die Spezifikation des SOAP-Encodings⁴⁰ regelt die Umwandlung von Daten in für beide Kommunikationspartner verständliches XML. Die Encoding-Regeln werden als XML-Schema im SOAP-Body mit dem Attribut *encodingStyle* eingebunden:

```
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
```

Die SOAP-Spezifikation stellt eine Vielzahl von Datentypen bereit, die in einer Nachricht verwendet werden können. Für das SOAP-Encoding stehen ähnliche Datentypen wie für den XML-Schema-Standard bereit. Eine genaue Auflistung der möglichen Datentypen und der unterschiedlichen Encoding-Formen finden sich in [SOAP].

5.1.4 Transportprotokolle

Das im obigen Beispiel eingesetzte HTTP-Transportprotokoll ist im Zusammenhang mit der SOAP-Nachrichten-Übermittlung am häufigsten vorzufinden. Die SOAP-Nachricht benötigt ein „Träger“-Transportprotokoll, welches die Nachricht zwischen Sender und Empfänger (quasi „verpackt“) weiterleitet. Seit Beginn der Web Service-Entwicklung nimmt HTTP einen bevorzugten Stellenwert ein. So enthält die SOAP-Spezifikation (Version 1.1) sogar einen eigenen Teil für den Einsatz von HTTP bei der SOAP-Nachrichten-Übermittlung. Mit HTTP geht SOAP dabei eine enge Bindung ein.

HTTP ist jedoch nicht zwingend das einzig nutzbare Transportprotokoll. Die SOAP 1.2 Spezifikation beinhaltet ein Rahmenwerk (SOAP Protocol Bindung Framework), welches das Einbinden anderer Transportprotokolle regelt, z. B. kann dafür auch das *Simple Mail Transport Protocol (SMTP)* eingesetzt werden:⁴¹

³⁹ Siehe [HL04] S. 52 f.

⁴⁰ <http://www.w3.org/2003/05/soap-encoding>

⁴¹ Vgl. hierzu [HL04] S. 65 f.

SMTP dient als Protokoll für den Austausch von elektronischer Post (E-Mails). In Verbindung mit SOAP ist jedoch nur das Absetzen einer Anfrage an den Web Service-Anbieter möglich, wobei die Antwort meist aus einer kurzen Bestätigung besteht. Die Anwendung „richtiger“ RPCs ist hiermit nicht möglich. Dies kann dadurch umgangen werden, dass der Serviceanbieter eine von dem Konsumenten erhaltene Anfrage (Request) so interpretiert, dass er daraufhin ebenfalls eine Anfrage an den Konsumenten sendet, die inhaltlich einer komplexeren Antwort auf die zuvor gestellten Anfrage darstellt. Mit diesem „Kunstgriff“ wird die SMTP-Beschränkung umgangen. In der Praxis wird SMTP meist dort eingesetzt, wo die Netzwerkphilosophie einer Firma (z. B. aus Sicherheitsgründen) kein HTTP mit der „Außenwelt“ erlaubt, jedoch den E-Mail-Verkehr über SMTP gestattet.

5.2 Googles Web Service

Google, als einer der beliebtesten Suchdienste im Web, bietet seit April 2002 einen eigenen Web Service⁴² an, der die Abfrage der Suchmaschine per Software erlaubt. Der Web Service befindet sich seit Beginn an im Testbetrieb (Beta), d. h. der Dienst wird derzeit von Google als experimentell eingestuft. Im August 2002 folgte eine aktualisierte Beta2 Version, die bis heute die aktuelle Version darstellt. Wie die Zukunft dieses Web Services aussehen wird, ob er die Betaphase überwindet, oder ob der Service vielleicht plötzlich wieder eingestellt wird, ist heute noch nicht abzusehen. Die GoogleAPI ist für neue, besondere, kreative Anwendungen geschaffen worden. Einige Vorschläge von Google sollen dabei die Phantasie der Entwickler anregen: automatisches Monitoring des Web für neue Informationen zu einem Subjekt, Marktforschungs-Tools und Trendanalysen, eine neue Benutzeroberfläche für die Suche etc.

Google stellt für seinen Web Service der Entwicklergemeinschaft ein API Developer Kit (*application program interface*) bereit. Im SDK⁴³ findet sich eine Kurzdokumentation in Form eines HTML-Dokuments, eine WSDL Datei (*GoogleSearch.wsdl*) für Verwendung in beliebigen Programmiersprachen sowie fertige Beispiele (und Klassen) für *Java* und *.NET*. Theoretisch kann die Google-Abfrage in jedes Programm eingebunden werden. Das Entwickler-Kit beinhaltet alles, was zum Schreiben eigener Programme, die die Google Web APIs⁴⁴ nutzen, benötigt wird.

5.2.1 Funktionalitäten

Ein Blick in *GoogleSearch.wsdl* (Listing 9) zeigt eine Zusammenfassung der Dienste im Zweig *portType*, wo derzeit drei Elemente *operation* existieren, die für die Nachrichten die Ein- und Ausgabefunktionen definieren. Es handelt sich dabei um die Google Suchfunktionen, die mit diesen Web Service angeboten werden.

⁴² <http://www.google.com/apis/index.html>

⁴³ SDK = Software Development Kit

⁴⁴ Die Bezeichnungen *Google Web Service* und *Google Web APIs Services* haben hier die gleiche Bedeutung.

```

1  (...)
2  <!-- Port for Google Web APIs, "GoogleSearch" -->
3  <portType name="GoogleSearchPort">
4      <operation name="doGetCachedPage">
5          <input message="typens:doGetCachedPage"/>
6          <output message="typens:doGetCachedPageResponse"/>
7      </operation>
8      <operation name="doSpellingSuggestion">
9          <input message="typens:doSpellingSuggestion"/>
10         <output message="typens:doSpellingSuggestionResponse"/>
11     </operation>
12     <operation name="doGoogleSearch">
13         <input message="typens:doGoogleSearch"/>
14         <output message="typens:doGoogleSearchResponse"/>
15     </operation>
16 </portType>
17 (...)

```

Listing 9: Auszug aus der WSDL-Datei.

Cache Request: Die Cache-Anfrage *doGetCachedPage* übermittelt einen URL und erhält als Ergebnis den in Base64⁴⁵ kodierten Inhalt des zugehörigen Webdokumentes.⁴⁶ Erfolg hat diese Anfrage nur, wenn Google für den URL den Inhalt in seinem Cache bereithält.

Spelling Request: Die Schreibweise-Anfrage *doSpellingSuggestion* übergibt eine Zeichenkette an den Google Web APIs Service und liefert – wenn verfügbar – ein Korrekturvorschlag bezüglich der Schreibweise, so wie es von der Google-Webseite bekannt ist.

Search Request: Die Such-Anfrage *doGoogleSearch* stellt die Standardsuche dar. Es wird hierfür ein Anfragestring zusammen mit weiteren Parametern (siehe Tabelle 1) übermittelt. Als Ergebnis werden die gefundenen Treffer in strukturierter Form – analog der Suche auf **www.google.com** – zurückgegeben.

5.2.2 Nutzungsbedingungen und Einschränkungen

Um den Google Web Service nutzen zu können, muss zuvor eine Registrierung bei Google erfolgen.⁴⁷ Im Gegenzug wird ein gültiger Account angelegt und der Nutzer erhält per E-Mail seinen eigenen Lizenzschlüssel, der zukünftig bei jeder Suchanfrage mit übersendet werden muss. Die Nutzung der Google Web APIs ist jedoch mit einigen Restriktionen verbunden:

- ☐ Die Anzahl der gestellten Suchanfragen ist auf **1000** pro Tag beschränkt.
- ☐ Pro Suchanfrage werden max. **10** Resultate zurückgeliefert.
- ☐ Der Anfragestring ist limitiert auf **2048** Bytes und max. **10** einzelne Wörter.

⁴⁵ Siehe [MS04] S. 593.

⁴⁶ Zu beachten ist hier, dass Google aus Gründen der Performance nur die ersten 110 KB einer Webresource ausliest und in seinem Cache speichert.

⁴⁷ Den Link zur Online-Registrierung findet sich unter: <http://www.google.com/apis/index.html>

- ❑ Der max. zulässige Wert von *start* + *maxResult* beträgt **1000**.
- ❑ Pro Anfrage darf der *site:-*Term nur **einmal** verwendet werden.

Über diese technischen Einschränkungen hinaus, gelten für den Web Service spezielle Nutzungsrichtlinien.⁴⁸ So ist z. B. ausdrücklich nur der Einsatz für einen persönlichen und nicht kommerziellen Gebrauch gestattet.

Tabelle 1: Parameter für die Google Anfrage

Parameter	Bedeutung
key	Lizenzschlüssel
q	Abfragestring
start	Start-Index des ersten Ergebnisses
maxResult	Anzahl der Ergebnisse
filter	Filteranweisung
restrict	Einschränkung der Suche
safeSearch	TRUE, wenn die Ergebnisse auch für Kinder geeignet sein sollen
lr	Einschränkung der Sprache

5.2.3 SOAP-Nachrichtenaustausch

Beispielhaft soll an dieser Stelle für die Google-Suchfunktion *doGoogleSearch* ein kompletter SOAP-Nachrichtenaustausch dargestellt werden. Dieses und weitere Beispiele werden mit dem Google APIs Developer's Kit bereitgestellt.

```

1 <?xml version='1.0' encoding='UTF-8'?>
2
3 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
  envelope/"
4   xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
5   xmlns:xsd="http://www.w3.org/1999/XMLSchema">
6   <SOAP-ENV:Body>
7     <ns1:doGoogleSearch xmlns:ns1="urn:GoogleSearch"
8       SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/
        encoding/">
9       <key xsi:type="xsd:string">*****</key>
10      <q xsi:type="xsd:string">Webtechnologien site:uni-jena.de</q>
11      <start xsi:type="xsd:int">0</start>
12      <maxResults xsi:type="xsd:int">10</maxResults>
13      <filter xsi:type="xsd:boolean">true</filter>
14      <restrict xsi:type="xsd:string"></restrict>
15      <safeSearch xsi:type="xsd:boolean">false</safeSearch>
16      <lr xsi:type="xsd:string"></lr>
17      <ie xsi:type="xsd:string">latin1</ie>
18      <oe xsi:type="xsd:string">latin1</oe>

```

⁴⁸ Siehe http://www.google.com/apis/api_terms.html

```

19     </ns1:doGoogleSearch>
20   </SOAP-ENV:Body>
21 </SOAP-ENV:Envelope>

```

Listing 10: *doGoogleSearch* (SOAP-Request)

```

1  <?xml version='1.0' encoding='UTF-8'?>
2  <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
   envelope/" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/1999/XMLSchema">
3    <SOAP-ENV:Body>
4      <ns1:doGoogleSearchResponse xmlns:ns1="urn:GoogleSearch" SOAP-
        ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
5        <return xsi:type="ns1:GoogleSearchResult">
6          <documentFiltering xsi:type="xsd:boolean">false</
            documentFiltering>
7          <estimatedTotalResultsCount xsi:type="xsd:int">3</
            estimatedTotalResultsCount>
8          <directoryCategories xmlns:ns2="http://schemas.xmlsoap.org/soap/
                encoding/" xsi:type="ns2:Array" ns2:arrayType="
                ns1:DirectoryCategory[0]"></directoryCategories>
9          <searchTime xsi:type="xsd:double">0.194871</searchTime>
10         <resultElements xmlns:ns3="http://schemas.xmlsoap.org/soap/
                encoding/" xsi:type="ns3:Array" ns3:arrayType="
                ns1:ResultElement[3]">
11           <item xsi:type="ns1:ResultElement">
12             <URL xsi:type="xsd:string">http://www.informatik.uni-jena.de
                /~sack/WS0405/webtechnologien-themen.htm</URL>
13             <cachedSize xsi:type="xsd:string">17k</cachedSize>
14             <directoryTitle xsi:type="xsd:string"></directoryTitle>
15             <hostName xsi:type="xsd:string"></hostName>
16             <relatedInformationPresent xsi:type="xsd:boolean">true</
                relatedInformationPresent>
17             <snippet xsi:type="xsd:string">Einführung, Themenübersicht
                und Materialien zur Vorlesung.</snippet>
18             <summary xsi:type="xsd:string"></summary>
19             <title xsi:type="xsd:string">Vorlesung: &lt;b>
                Webtechnologien&lt;/b></title>
20           </item>
21           (...)
22         </resultElements>
23         <endIndex xsi:type="xsd:int">3</endIndex>
24         <searchTips xsi:type="xsd:string"></searchTips>
25         <searchTime xsi:type="xsd:double">0.122631</searchTime>
26         <startIndex xsi:type="xsd:int">1</startIndex>
27         <estimateIsExact xsi:type="xsd:boolean">true</estimateIsExact>
28         <searchQuery xsi:type="xsd:string">Webtechnologien site:uni-
                jena.de</searchQuery>
29       </return>
30     </ns1:doGoogleSearchResponse>
31   </SOAP-ENV:Body>
32 </SOAP-ENV:Envelope>

```

Listing 11: *doGoogleSearch* (SOAP-Response)

Teil II

Implementierung

6 Vorgehensweise

Im Folgenden werden die für die Lösung der Aufgabenstellung eingesetzten Methoden Schritt für Schritt erarbeitet und teils an Abbildungen und Beispielen beschrieben. Die aufgetretenen Schwierigkeiten werden nicht verschwiegen und bilden an einigen Stellen die Grundlage für die Begründung des eingeschlagenen Lösungsweges.

Nachdem im nächsten Absatz die Wahl der eingesetzten Programmiersprache begründet wird, beginnt die Beschreibung der Implementierung mit der Erläuterung der Beschaffung einer Datengrundlage. Dies beinhaltet die Umsetzung eines SOAP-Clients, der für die Kommunikation mit dem Google Web Service benötigt wird. Ein erarbeitetes Architekturmodell illustriert die einzelnen Systemkomponenten im Hinblick auf den Ort der jeweiligen Umsetzung und ihrer Aufgaben. Es folgt eine Beschreibung der durchlaufenen Arbeitsschritte für die Ontologieerstellung. Weiter wird gezeigt, wie die Informationen aus der Ontologie, mittels eines Parsers, in eine geeignete Datenstruktur für die anschließende Nutzung überführt werden. Die Verbindung zwischen Ontologie und der Datengrundlage einer Suchmaschine wird Schritt für Schritt anhand einzelner Teile der fertigen Implementierung vorgestellt. Es wird gezeigt, welche Ideen hinter den einzelnen logischen Teilen der Implementierung stecken und wie sie realisiert wurden. Alle Auszüge aus dem Quelltext geben meist nur den Kern der umgesetzten Funktionsweise wieder und wurden von programmspezifischen Besonderheiten und Kommentaren bereinigt. Die wirkliche Implementierung weist eine komplexere Syntax auf, auf die hier jedoch zu Gunsten der Lesbarkeit verzichtet wird.

Der im Praxisteil implementierten Software wurde der Name **Sonto^X** (Search **Ontologie** e**XTension**) verliehen, synonym stehend für die umgesetzte Konzeption.⁴⁹ Die entwickelte Websuche ist im WWW unter dem URL: <http://www.artusweb.de/SontoX> zu erreichen.

Die Wahl der Programmiersprache

Vor Beginn der Programmierung einer Web-Applikation muss sich für eine geeignete Programmiersprache entschieden werden. Die Applikation selbst soll später auf einem Server – dem Web-Server – im WWW erreichbar sein und dort mit der *Benutzerschnittstelle*

⁴⁹ Der Name *Sonto^X* (sprich: „sontox“) ist mehr ein Kunstname als ein Akronym und setzt sich aus Search, *ontology* und e**XTension** zusammen. Damit soll das Konzept ausgedrückt werden, mit Hilfe einer Ontologie-Erweiterung die Suchtreffer einer konventionellen Suchmaschine möglichst mit zusätzlichen (semantischen) Informationen zu versehen.

(Web-Interface) interagieren. Mit Hilfe des *Common Gateway Interfaces (CGI)*⁵⁰ ist es möglich, eine externe Anwendung für die Erstellung dynamischer HTML-Dokumente, auf Basis der Benutzereingaben und in Abhängigkeit der gebotenen Funktionalität der Anwendung, zu verwenden. Diese Anwendung kann z. B. in einer Hochsprache wie Java, C oder C++ geschrieben werden. Wer jedoch nicht gerade einen WWW-Server sein Eigen nennen darf, bzw. erweiterte administrative Rechte für einen solchen besitzt, ist auf einen dementsprechenden Serviceanbieter – einen sog. Host-Provider – angewiesen. Nicht zuletzt aus Sicherheitsgründen verbieten oder beschränken die Provider ihren Kunden zumeist die Ausführung eigener ausführbarer Programme auf ihren Servern. In der Regel stehen aber populäre Skriptsprachen wie z. B. Perl oder PHP zur Verfügung.

Bei PHP (*PHP Hypertext Preprocessor*)⁵¹ handelt es sich um eine serverseitig ausgeführte Web-Skriptsprache, mit deren Hilfe eine schnelle und effiziente Entwicklung dynamischer Webanwendungen ermöglicht wird. PHP ist eine schnelle, umfangreiche und leistungsstarke Skriptsprache und besticht vor allem durch seinen großen Funktionsumfang. Die Sprache bietet die verbreitete C-Syntax, gute Modellierungsmöglichkeiten und kann dabei als plattformunabhängig angesehen werden, da sie für alle gängigen Plattformen bereitsteht.

Trotz der vielen Vorzüge ist PHP5 jedoch eher zu den semiprofessionellen Sprachen zu zählen. So gibt es eine Vielzahl von typischen objektorientierten Eigenschaften, die in PHP5 noch nicht umgesetzt wurden.⁵² Weiterhin ist und bleibt PHP *nur* eine Skriptsprache, bei der für die Ausführung eines Programms ein Parser – der *Preprocessor* – zum Einsatz kommt. Dies hat einen negativen Einfluss auf die Performance einer Anwendung. Aufgrund der Notwendigkeit des Parsens für jeden einzelnen Programmaufruf, kommt es zu einem gewissen Mehraufwand (*Grund-Over-Head*).

Es bleibt zusammenfassend festzuhalten, dass PHP trotz seiner Einschränkungen, für kleinere und mittlere Web-Projekte trotzdem hervorragend geeignet ist. Da Webanwendungen untrennbar mit HTML zusammenarbeiten, ist PHP schon deshalb eine gute Wahl. Der *Grund-Over-Head* führt bei kleineren und mittleren PHP-Anwendungen zu keiner spürbaren Performanceeinbuße. Für die Implementierung wurde sich daher für die Programmiersprache *PHP* in der Version 5 entschieden. Dabei wurde auf eine möglichst konsequente objektorientierte Umsetzung des gesamten *Sonto*^x-Quelltextes geachtet, um eine eventuelle spätere Implementierung, in eine andere Programmiersprache, weitestgehend zu erleichtern.

7 Beziehen der Datengrundlage

Der Ausgangspunkt für die Beschaffung einer Datengrundlage liegt in einer Suchabfrage bei einer der vielzähligen im Web vertretenen Suchdienste. Als Datengrundlage wird der Datenbestand einer Suchmaschine verstanden und zwar speziell der Teil, der bei einer Suchanfrage dem Nutzer im Web-Interface der Suchmaschine angezeigt wird. Bei der

⁵⁰ Siehe [MS04] S. 1079 ff.

⁵¹ Projekthomepage von PHP: <http://www.php.net>

⁵² Vgl. [Kra04] S. 25 und S. 272 ff.

Nutzung, der durch eine Suchmaschine indizierten Datenbasis, ergeben sich erhebliche Erleichterungen für die Weiterverarbeitung der Daten. Konkret kann von Folgendem ausgegangen werden:

- ❑ Der URL eines Treffers ist *gültig*, d. h. die URL-Syntax entspricht der vereinbarten Norm.⁵³ Eine Überprüfung der URLs (*Validierung*) auf ihre syntaktische Korrektheit hin (vor einer Weiterverarbeitung) kann daher entfallen.
- ❑ Die *URL-Existenz* ist mit großer Wahrscheinlichkeit gegeben, da alle erhaltenen URLs, von der Suchmaschine vor kurzem erfolgreich auf Erreichbarkeit geprüft wurden.

Sonto^x wurde so konzipiert, dass die Wahl des eigentlichen Verfahrens zur Suchmaschinenabfrage möglichst unabhängig vom Kernsystem realisiert ist. Dies hat den Vorteil, dass bei Bedarf relativ unkompliziert ein Wechsel zu einem anderen Verfahren erfolgen kann. Dafür wurde die Funktion der Datenbeschaffung durch eine eigene Anfrage-Klasse (*INQUIRY.class*) (siehe Abb. 11) realisiert. Bei einem Wechsel zu einer anderen Suchmaschine, muss lediglich die Anfrage-Klasse modifiziert werden, in der gegebenenfalls weitere Klassen, die spezielle auf die gewählte Suchmaschine abgestimmte Methoden bereit halten, einzubinden sind.

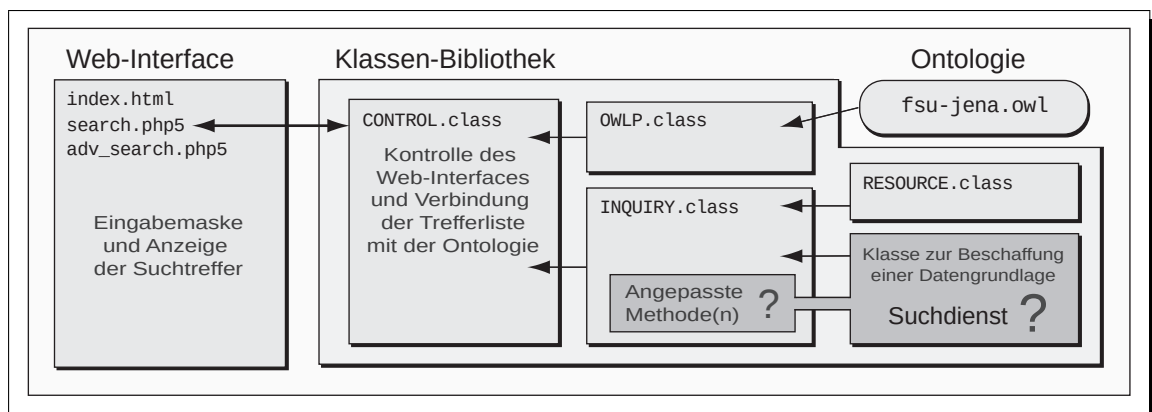


Abbildung 11: Architektur-Modell mit gekennzeichneten Schnittstelle zur Datenbeschaffung.

Abbildung 11 zeigt die *Sonto*^x-Systemarchitektur⁵⁴ in der die zur Datenbeschaffung relevanten Programmteile dunkelgrau markiert sind.

In den nächsten beiden Teilabschnitten werden zwei grundsätzliche Methoden zur Lösung des Datenproblems vorgestellt.

7.1 Methode des Screen Scrapings

Eine Möglichkeit der Datengewinnung besteht darin, ein HTTP-Request an eine Suchmaschine seiner Wahl zu senden und die durch HTTP-Response zurück gelieferte Treffersei-

⁵³ Die Spezifikation der URL-Syntax ist in RFC 1738 und 1808 standardisiert.

⁵⁴ Abbildung 11 zeigt eine vereinfachte Variante des in Anhang B auf Seite 79 aufgeführten detaillierteren *Sonto*^x-Architekturmodells.

te anhand von HTML-Analyseverfahren so zu parsen, dass die enthaltenen Informationen in einer für die Aufgabenstellung strukturierten Form vorliegen. Dieses Vorgehen wird als *Screen Scraping* [Fer03] bezeichnet und bietet eine hohe Flexibilität in der Benutzung und Anpassung auf individuelle Aufgabenstellungen. Wer sich für diese Lösung der Datengewinnung entscheidet, muss jedoch zwei grundlegende Einschränkungen beachten. Zum einen muss bei einer Änderung – welche zuerst als solche erkannt werden muss – der Webseitenstruktur des Contentanbieters die Screen-Scraping-Software auf die neuen Gegebenheiten hin angepasst werden. Dies kann von Fall zu Fall sehr aufwendig sein und viel Zeit in Anspruch nehmen, wobei ein Erfolg nicht garantiert ist. Zum anderen sehen es einige Suchmaschinenbetreiber nicht gerne, wenn automatische Anfragen an ihr Webinterface gestellt werden. So weist z. B. Google in seinen Dienstleistungsbedingungen⁵⁵ darauf hin, dass keine „Meta-Suche“ per Software erlaubt ist. Hierzu müsste ein Zusatzvertrag mit Google vereinbart werden. Um unliebsame Überraschungen zu vermeiden, sollte daher zuvor ein gründlicher Blick in die Nutzungsbedingungen des jeweiligen Suchdienstes geworfen werden.

Obwohl in *Sonto*^x ein anderes Verfahren zum Einsatz kommt, soll die Option des Screen Scrapings – als zweite Wahl – nicht für eine spätere alternative Realisierung der Datenbeschaffung aus den Augen verloren werden, weil dadurch ein breites Spektrum von Suchmaschinen genutzt werden kann.

7.2 Nutzung des Googles Web Services

Eine weitaus elegantere und zuverlässigere Methode zur Datengewinnung stellt der im Teil I vorgestellte Google Web Service dar. Er ermöglicht eine bequeme und relativ einfache Möglichkeit einer WWW-Suche aus einem Anwendungsprogramm heraus.

Bedenklich sind jedoch die von Google auferlegten Beschränkungen. Dass nur max. zehn Suchergebnisse pro Anfrage zurückgegeben werden, ist durch im Hintergrund ablaufende Mehrfachabfragen leicht kompensierbar. Auch kann durch die Angabe eines exakten Startindex (0 bis max. 1000), der Anwendungsprogrammierer eine entsprechende Ergebnisdekade⁵⁶ anfordern und dadurch bequem durch die Trefferliste navigieren. Hierzu muss jedoch bei der Umsetzung der Programm-Logik etwas Mühe investiert werden, um eine exakte Navigation durch die einzelnen Suchtrefferseiten umzusetzen.

⁵⁵ <http://www.google.de/intl/de/terms.html>

⁵⁶ Gemeint sind die jeweils zehn erlaubten Suchtreffer aus der Menge aller möglichen gefundenen Treffer pro Anfrage an den Google-Server.

Die zweite wichtige Einschränkung erfordert hingegen eine prinzipielle Diskussion über das *Für* und *Wider* des eingesetzten Web Services. Die Rede ist von der Beschränkung auf max. 1000 Einzelabfragen pro Tag. Stellt diese Schranke für eine kleine private Homepage kaum ein Problem dar, so ist bei größeren Webseiten-Projekten die Grenze eventuell schnell erreicht. Ausschlaggebend sind hierfür die Anzahl der sog. *Klicks*, d.h. die täglichen gestellten Anfragen. Es muss also genau im Vorfeld analysiert werden, welcher Zielgruppe der eigene Suchdienst angeboten werden soll und ob eine tägliche Nutzerzahl jenseits der 1000er Marke zu erwarten ist.

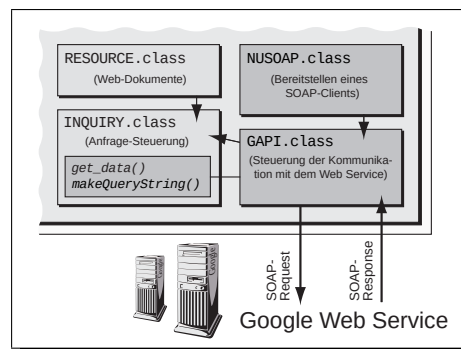


Abbildung 12: Google API Service als Datengrundlage

Nichtsdestotrotz erscheint der Google Web Service äußerst verlockend, sodass sich trotz Beschränkungen für dessen Einsatz entschieden wurde. Obwohl in dieser Arbeit schlussendlich eine für alle im Netz verfügbare *Sonto*^x-Version angestrebt wird, kann nur der Praxiseinsatz zeigen, ob die 1000er Marke überschritten wird.

Abbildung 12 zeigt als konkrete Realisierung einer Datenbeschaffung die Integration des Google Web Services in die schon zuvor vorgestellte Systemarchitektur. Die dunkelgrau unterlegten Felder sind die speziell auf den Google Web Service zugeschnittenen Systemkomponenten.

Implementierung eines SOAP-Clients

Um die Schnittstelle zu Google nutzen zu können, ist die Implementierung eines SOAP-Client erforderlich. Der SOAP-Client ist ein Teil des Gesamtsystems, der die Kommunikation mit dem Web Service übernimmt. PHP hält ab der Version 5 die Möglichkeit eines internen SOAP-Moduls bereit. Unter Linux muss dazu z. B. PHP5 mit dem Schalter *-with-soap* kompiliert werden.⁵⁷ Danach ist es möglich, die fest implementierte SOAP-Klasse für die Erstellung eines eigenen SOAP-Clients zu nutzen.⁵⁸ Da für *Sonto*^x eine möglichst große Unabhängigkeit von einer speziellen Webserverkonfiguration angestrebt wurde und die meisten Host-Provider PHP5 ohne eine SOAP-Unterstützung bereitstellen, empfiehlt sich die Umsetzung eines eigenen SOAP-Clients.

Obwohl das SOAP-Protokoll relativ einfach aufgebaut ist und das XML-Austauschformat eine Kontrolle des Nachrichtenaustausches vereinfacht, sollte der Aufwand für die Umsetzung eines eigenen SOAP-Clients nicht unterschätzt werden. Die Entwicklergemeinschaft von PHP hält speziell für die Implementierung eines SOAP-Clients einige frei zugängliche Klassen bereit, sodass eine eigene Programmierung nicht notwendig ist. In *Sonto*^x wurde

⁵⁷ Siehe hierzu <http://www.php.net>

⁵⁸ Vgl. dazu [Kra04].

hierfür das Open-Source-Projekt *NuSOAP*⁵⁹ verwendet.⁶⁰ Es handelt sich dabei um eine Kollektion von PHP-Klassen, welche eine rasche Umsetzung eines SOAP-Clients ermöglichen.

Listing 12 zeigt die Stellen in der eigens erstellten Klasse zur Steuerung mit der Google API (*GAPI*-Klasse), an denen die *NUSOAP*-Klasse zum Einsatz kommt.⁶¹

```

1  (...)
2  // Einbinden der NuSOAP-Class Version 0.6.3 von D. Ayala.
3  include('NUSOAP.class.php');
4
5  // Instanzieren eines SOAP-Client Objektes.
6  $soapclient = (object) new soapclient('http://api.google.com/search/
    beta2');
7  (...)
8  // Google-Anfrage über die doGoogleSearch-Methode
9  $this->myResult = $soapclient->call('doGoogleSearch',(array) $this->
    Params, 'urn:GoogleSearch', 'urn:GoogleSearch');
10 (...)

```

Listing 12: SOAP-Client unter Verwendung der *NUSOAP*-Klasse.

Nach dem Einbinden der *NUSOAP*-Klasse (Zeile 3) stehen alle für einen Client benötigten Methoden zur Verfügung. Der Aufruf des Konstruktors mit dem URL des Nachrichtempfängers (***http://api.google.com/search/beta2***) als Parameter, erzeugt das gewünschte Client-Objekt (Zeile 6). Nach erfolgreicher Instanzierung des SOAP-Clients wird über die Objekt-Variable *\$soapclient* die PHP *call*-Methode aufgerufen (Zeile 9). Als Parameter wird der Name der gewünschten Web Service-Funktion (hier *doGoogleSearch*) und ein spezieller Parametersatz (*\$this->Params*) übergeben. Ein Blick in die WSDL-Datei zeigt eine Zusammenstellung aller in *\$this->Params* (Zeile 9) benötigten Parameter mit ihren jeweiligen Datentypen (Listing 13). Die Bedeutung der einzelnen Parameter wird in [Google] genau spezifiziert und wurde in Tabelle 1 auf S. 35 bereits kurz vorgestellt. Nach einer erfolgreichen Anfrage, enthält die Variable *\$this->myResult* die Ergebnisinformationen in Form eines assoziativen Arrays.

```

1  (...)
2  <message name="doGoogleSearch">
3    <part name="key"           type="xsd:string"/>
4    <part name="q"             type="xsd:string"/>
5    <part name="start"          type="xsd:int"/>
6    <part name="maxResults"      type="xsd:int"/>
7    <part name="filter"          type="xsd:boolean"/>
8    <part name="restrict"        type="xsd:string"/>
9    <part name="safeSearch"      type="xsd:boolean"/>
10   <part name="lr"              type="xsd:string"/>

```

⁵⁹ NuSOAP ist ein *Web Services Toolkit for PHP* und wird unter der *GNU Lesser General Public License* von der *NuSphere Corporation* bereitgestellt: <http://www.nusphere.com>.

⁶⁰ Alternativ sei auf die *PEAR*-Klassenbibliothek verwiesen, die ebenfalls einige Klassen zum Thema bereitstellt: <http://pear.php.net>.

⁶¹ Die Zeilennummerierung dient – hier und im Folgendem – der Referenzierung einer entsprechenden Zeile aus dem Text heraus und entspricht *nicht* der originalen Nummerierung des Quelltextes.

```

11     <part name="ie"                type="xsd:string"/>
12     <part name="oe"                type="xsd:string"/>
13 </message>
14 (...)

```

Listing 13: Parameter für *doGoogleSearch* (WSDL-Datei).

Das assoziative Array beinhaltet nach jeder Suchanfrage einen Satz von Meta-Daten als Statusinformation (Listing 14, Zeilen 2–16) und ressourcenspezifische Informationen für jeden einzelnen der zehn Suchtreffer (Zeilen 18–30). Die genaue Semantik dieser Variablen wird ebenfalls in [Google] spezifiziert. Prinzipiell wird dadurch die identische Information, die auch bei einer Suche über das Google-Web-Interface geboten wird, bereitgestellt.

```

1  (...)
2  <xsd:complexType name="GoogleSearchResult">
3    <xsd:all>
4      <xsd:element name="documentFiltering" type="xsd:boolean"/>
5      <xsd:element name="searchComments" type="xsd:string"/>
6      <xsd:element name="estimatedTotalResultsCount" type="xsd:int"/>
7      <xsd:element name="estimateIsExact" type="xsd:boolean"/>
8      <xsd:element name="resultElements" type="typens:ResultElementArray"
9        "/>
10     <xsd:element name="searchQuery" type="xsd:string"/>
11     <xsd:element name="startIndex" type="xsd:int"/>
12     <xsd:element name="endIndex" type="xsd:int"/>
13     <xsd:element name="searchTips" type="xsd:string"/>
14     <xsd:element name="directoryCategories" type="
15       typens:DirectoryCategoryArray"/>
16     <xsd:element name="searchTime" type="xsd:double"/>
17   </xsd:all>
18 </xsd:complexType>
19
20 <xsd:complexType name="ResultElement">
21   <xsd:all>
22     <xsd:element name="summary" type="xsd:string"/>
23     <xsd:element name="URL" type="xsd:string"/>
24     <xsd:element name="snippet" type="xsd:string"/>
25     <xsd:element name="title" type="xsd:string"/>
26     <xsd:element name="cachedSize" type="xsd:string"/>
27     <xsd:element name="relatedInformationPresent" type="xsd:boolean"/>
28     <xsd:element name="hostName" type="xsd:string"/>
29     <xsd:element name="directoryCategory" type="
30       typens:DirectoryCategory"/>
31     <xsd:element name="directoryTitle" type="xsd:string"/>
32   </xsd:all>
33 </xsd:complexType>
34 (...)

```

Listing 14: Datentypen *GoogleSearchResult* und *ResultElement* (WSDL-Datei).

7.3 Anfragesteuerung in einer eigenen Anfrage-Klasse

Die benötigte Anfragesteuerung zur Erhaltung einer Datengrundlage wird in einer eigens definierten Anfrage-Klasse (*INQUIRY.class*) zusammengefasst (siehe Abb. 12). Die Klasse selbst ist in der Hauptklasse (*CONTROL.class*) eingebunden und wird von dort aus instanziiert. Der Aufruf der Methode *get_data()*, einer zuvor erzeugten Objektinstanz der *INQUIRY*-Klasse, löst eine Anfrage an eine Suchmaschine aus.

Dazu kann prinzipiell jede beliebige Suchmaschine abgefragt werden. Die Anfrage wird automatisch bei der Instanzierung eines *CONTROL*-Objektes in der *CONTROL*-Klasse vorgenommen, sodass ein eventuelles Anfrageergebnis bereits kurz nach Beginn der Programmausführung vorliegt.

In der *INQUIRY*-Klasse selbst wird für jedes *ResultElement* ein *RESOURCE*-Objekt generiert.⁶² Nach Aufruf der *get_data()*-Methode aus der *INQUIRY*-Klasse heraus, besitzt das *INQUIRY*-Objekt die für die weitere Verarbeitung wichtigen Eigenschaften *\$Resources* und *\$ResponseMetaData*. Die Variable *\$Resources* ist dabei ein Array von Objektinstanzen der *RESOURCE*-Klasse und die Variable *\$ResponseMetaData* enthält, in Form eines assoziativen Arrays, die wichtigsten Metadaten der Suchanfrage.

```

1  (...)
2  // Einbinden der RESOURCE-Klasse: Jedes Webdokument der Trefferliste
3  // wird als Objektinstanz der RESOURCE-Klasse repräsentiert.
4  include('RESOURCE.class.php');
5  (...)
6  $i=0;
7  foreach ($GAPI->myResult['resultElements'] as $r) {
8      // neues Resource-Objekt erzeugen (-> Array von Resource-Objekten)
9      $this->Resources[$i] = (object) new RESOURCE((string) $r['URL']);
10     $this->Resources[$i]->title = $r['title'];
11     $this->Resources[$i]->snippet = $r['snippet'];
12     $this->Resources[$i]->cachedSize = $r['cachedSize'];
13     $i++;
14 }
15 (...)

```

Listing 15: Verarbeiten der Suchergebnisse in der *INQUIRY*-Klasse.

Listing 15 zeigt, dass nicht das volle Potenzial an Ressourceninformationen in *Sonto*^x genutzt wird. Momentan werden nur der Titel (*title*), die kurze Beschreibung (*snippet*) und die Größe des im Cache gehaltenen Dokumententeiles (*cachedSize*) verwendet⁶³, da diese für fast alle Treffer zurückgegeben werden und für eine Beschreibung einer Webressource ausreichend sind. Bei Bedarf können weitere Result-Elemente aufgenommen werden, wie z. B. der Host-Name (*hostName*) oder die Kategorie der von Google geführten *Open Directory Categories* (*directoryCategory*).⁶⁴

⁶² Siehe Listing 15.

⁶³ Der URL wird schon bei der Instanzierung ebenfalls mit verwendet.

⁶⁴ Siehe [Google].

8 Erstellen der Ontologie

Bevor die Suchtreffer mit mehr Semantik aufbereitet werden können, muss eine geeignete Ontologie für dieses spezielle Problem erstellt werden. In *Sonto*^x soll diese Ontologie die Basis der erweiterten Suche darstellen und das Verhalten der Web-Anwendung bestimmen.

In der Ontologie sollen diejenigen Informationen untergebracht werden, die für die Zuordnung einzelner Webressourcen zu einer Ebene der konkreten Universitätsstruktur benötigt werden. Z. B. soll eine Institutsseite als solche erkannt und einer Fakultät als übergeordnete Instanz zuordenbar sein. An dieser Stelle muss überlegt werden, welche Strukturen der FSU Jena sinnvoll für die Aufgabe sind und welche dafür weniger relevant sind. Die Überlegung führt zu dem Schluss, dass eine 1:1 Modellierung der kompletten und detaillierten Universitätsstruktur kaum der Aufgabe entspricht. Die Ontologie wäre damit zwar vollständig und universell einsetzbar, jedoch würde der Aufwand für die Modellierung und die Dateigröße selbst in keinem Verhältnis zu dem Nutzen für eine Webseitenerkennung stehen. Eine spezielle, für die Aufgabenstellung zugeschnittene Ontologie, in der nur solche Konzepte der Wirklichkeit modelliert werden, die auch von Nutzen sind, stellt hier daher den besseren Weg dar.

Zur Kodierung der Ontologie, kommt in *Sonto*^x, der vom W3C propagierte neue OWL-Sprachstandard zum Einsatz. Zu Beginn dieser Arbeit wurde versucht, die Ontologie manuell in einem einfachen Texteditor zu kodieren. Es sollte so ein möglichst nahes Arbeiten mit der OWL-Syntax erreicht werden. Die Basiskonstrukte wurden hierfür analog zur OWL-Referenz verwendet, um eine Standardkonformität zu gewährleisten. Der Autor sah sich jedoch schnell zwei Problemen gegenüber. Zum einen stellte sich die Frage der Validierung der Ontologie und zum anderen sollte es späteren Anwendern ermöglicht werden, die Ontologie leicht zu ändern oder zu erweitern, ohne im Detail mit der OWL-Sprachsyntax vertraut zu sein. Wie im Teil I der Arbeit vorgestellt, bietet sich hier Protégé als Entwicklungsumgebung an, welche unter anderem eine korrekte OWL-Syntax garantiert. Weiterhin kann sich der Entwickler voll und ganz auf die logische Ebene konzentrieren, ohne sich im Detail mit der Sprachreferenz auseinandersetzen zu müssen.

Die Modellierung der Ontologie wurde in dieser Arbeit mit Hilfe von Protégé in der Version 3.0 vorgenommen. Die reine Modellierung und die Wissensakquise gingen dabei Hand in Hand und ließen sich nicht klar trennen, da erst die Akquise der Daten selbst, die Schwächen in der Modellierung aufzeigte, woraufhin das Modell der Ontologie angepasst wurde. Die im Ergebnis entstandene Ontologie kristallisierte sich als am besten geeignet für die Problemstellung heraus. Die Erstellung wird in drei Schritte untergliedert:

- ① Festlegung einer geeigneten Menge von *Klassen*.
- ② Eine auf die Klassen bezogene Definition sinnvoller *Eigenschaften*.
- ③ Wissensakquise durch Erzeugung konkreter abgeleiteter *Klasseninstanzen*.

Im Folgenden wird das Wesentliche der Umsetzung ergebnisorientiert erläutert, ohne auf die konkrete Anwendung von Protégé näher einzugehen.⁶⁵ Weiter unten (in Abschnitt

⁶⁵ Für Protégé existiert ein ausführliches Benutzerhandbuch (<http://protege.stanford.edu/useit.html>).

8.5) finden sich wichtige Hinweise, die bei einer Ontologierstellung mit Protégé beachtet werden müssen, um eine mit dem *Sonto*^X-System kompatible Ontologie zu erhalten.

8.1 Festlegen der benötigten Klassen

Das Festlegen der benötigten Klassen stellt das Grundgerüst einer Ontologie-Definition dar. Die Identifizierung der Klassen ist die erste Aufgabe. Welche Klassen genau aufgenommen werden und wie detailliert die Struktur ausfallen soll, musste hierbei immer, mit Blick auf den Nutzen, abgewogen werden. Abbildung 13 zeigt die Klassen, die für die Arbeit letztendlich als relevant eingeschätzt wurden. Es sind dabei nur solche Entitäten als Klasse definiert, für die eine ausreichend große Anzahl an aussagekräftigen Webressourcen existiert.

Bei der Definition der Klassen darf nicht der Fehler begangen werden, die logischen Hierarchiestufen der Universität, anhand des *subClassOf*-Konstruktes in Protégé umzusetzen. Dafür dürfen ausschließlich und allein Gesichtspunkte, wie sie beim Klassenkonzept der objektorientierten Programmierung zum Einsatz kommen, angewandt werden. D. h. es muss überlegt werden, welche Eigenschaften die einzelnen relevanten Organisationsstrukturen haben und an welcher Stelle eine Klassenhierarchie bei der Definition sinnvoll erscheint. In Abbildung 13 gilt für die rechts eingerückten Klassen die *subClassOf*-Beziehung zu der Klasse *Organisation*, die die Wurzelklasse darstellt und eine Sub-Klasse von *owl:Thing* ist. Listing 16 zeigt einen Auszug, der von Protégé automatisch generierten Syntax der OWL-Klassendefinition.⁶⁶



Abbildung 13: Das Klassenkonzept der Ontologie.

```

1  (...)
2  <owl:Class rdf:ID="Arbeitsgruppe">
3    <rdfs:subClassOf>
4      <owl:Class rdf:ID="Organisation"/>
5    </rdfs:subClassOf>
6  </owl:Class>
7  <owl:Class rdf:ID="Institut">
8    <rdfs:subClassOf rdf:resource="#Organisation"/>
9  </owl:Class>
10 (...)

```

Listing 16: Auszug aus der Klassendefinition (fsu-jena.owl).

Wie in den Zeilen 2–7 im Listing 16 zu sehen ist, entscheidet Protégé von Fall zu Fall selbst, über die Art und Weise der Anwendung der OWL-Konstrukte. Die Definition der Klasse *Organisation* erfolgt an Ort und Stelle der eigentlichen *subClassOf*-Definition der Klasse *Arbeitsgruppe* (Zeile 4).

⁶⁶ Die komplette OWL-Ontologie ist unter dem URL <http://www.artusweb.de/SontoX/ontology/fsu-jena.owl> einzusehen.

8.2 Definition der benötigten Eigenschaften

Nach der Wahl der eingesetzten Klassen, wurde ein hinreichend großer Satz von zugehörigen Eigenschaften definiert. Gesucht waren nur solche Eigenschaften, die typisch und aussagekräftig für eine spätere Webseiten-Instanz sind. Dies ist z. B. bei der Eigenschaft *Homepage* gegeben, da der spätere Wert (die Angabe eines URL) eine Webressource eindeutig referenziert. In Tabelle 2 sind alle aufgenommenen Eigenschaften alphabetisch aufgelistet:

Tabelle 2: Definierte Eigenschaften der Ontologie.

Bezeichnung	OWL-Typ	Wertezuordnung
Anschrift	DatatypeProperty	<i>string</i> (XMLS Datatype)
Dekan	DatatypeProperty	<i>string</i> (XMLS Datatype)
Direktor	DatatypeProperty	<i>string</i> (XMLS Datatype)
E-Mail	DatatypeProperty	<i>string</i> (XMLS Datatype)
Fax	DatatypeProperty	<i>string</i> (XMLS Datatype)
gehört_zu	ObjectProperty	<i>instance of class</i>
Homepage	DatatypeProperty	<i>anyURL</i> (XMLS Datatype)
Inhaber	DatatypeProperty	<i>string</i> (XMLS Datatype)
Leiter	DatatypeProperty	<i>string</i> (XMLS Datatype)
Picturer	DatatypeProperty	<i>anyURL</i> (XMLS Datatype)
Telefon	DatatypeProperty	<i>string</i> (XMLS Datatype)

Die Entscheidung über den endgültigen Satz an Eigenschaften wurde – analog zur Klassenwahl – nach einigen Tests im Zusammenhang mit der Benutzerschnittstelle getroffen. Zwei Eigenschaften wurden dabei von Anfang an als Notwendigkeit für das *Sonto^X*-System identifiziert:

- ❑ die ObjectProperty *gehört_zu*,
- ❑ und die DatatypeProperty *Homepage*.

Beide spielen eine ausschlaggebende Rolle bei der gesamten Umsetzung und sind *zwingend notwendig*, um ein korrektes Arbeiten von *Sonto^X* zu gewährleisten.⁶⁷ Mit Hilfe von *gehört_zu* wird ein Aufbau einer Taxonomie, der noch zu akquirierenden Instanzen, überhaupt erst möglich. Die Eigenschaft fungiert als Bindeglied zur Referenzierung einer Instanz aus einer anderen heraus. In Abschnitt 10.4 wird dieser Sachverhalt näher erläutert. Über die zweite wichtige Eigenschaft *Homepage* wird eine eindeutige Zuordnung einer im Treffersatz der Suchmaschine aufgeführte Webseite zu einer Instanz aus der Ontologie sichergestellt. Auch dieses Konzept gehört zur Hauptidee, die hinter *Sonto^X* steckt.⁶⁸ Listing 17 zeigt für beide Eigenschaften einen Auszug der von Protégé erzeugte OWL-Syntax der Definition.

⁶⁷ Siehe Abschnitt 8.5, S. 50.

⁶⁸ Siehe Abschnitt 10.1, S. 58.

```

1  (...)
2  <owl:ObjectProperty rdf:ID="gehört_zu">
3    <rdfs:domain>
4      <owl:Class>
5        <owl:unionOf rdf:parseType="Collection">
6          <owl:Class rdf:about="#Fakultät"/>
7          (...)
8          <owl:Class rdf:about="#Klinik"/>
9          <owl:Class rdf:about="#Lehrstuhl"/>
10         </owl:unionOf>
11       </owl:Class>
12     </rdfs:domain>
13 </owl:ObjectProperty>
14 (...)
15 <owl:DatatypeProperty rdf:ID="Homepage">
16   <rdfs:domain rdf:resource="#Organisation"/>
17   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#anyURI"/>
18 </owl:DatatypeProperty>
19 (...)

```

Listing 17: Auszug der Definition für *gehört_zu* und *Homepage* (fsu-jena.owl).

In der Definition der *gehört_zu*-Eigenschaft wird das *unionOf*-Tag, in Kombination mit dem Attribut *parseType="Collection"*, zur zusammenfassenden Festlegung des gültigen Wertebereiches (*domain*) verwendet (siehe Listing 17, Zeilen 2–13). Für den Wertebereich der Eigenschaft *Homepage* wurde hingegen nur die Klasse *Organisation* angegeben (Zeile 16). Alle anderen Klassen erben diese Eigenschaft implizit aufgrund der *subClassOf*-Beziehung zur Klasse *Organisation*. Als Datentyp wird *anyURI* aus der XMLSchema-Definition festgelegt (Zeile 17).

8.3 Wissensakquise – Das Schaffen einer Wissensbasis

Auf Grundlage der zuvor definierten Klassen und Eigenschaften wird eine Wissensbasis aufgebaut. Dieser Vorgang wird auch als Wissensakquise bezeichnet. Voraussetzung war hierfür eine umfangreiche Recherche der aktuellen „Webseitenlandschaft“ der FSU Jena. Es wurden nur solche Organisationseinheiten der FSU Jena aufgenommen, für die eine hinreichend große Webpräsenz angeboten wird. Die Identifizierung erfolgte manuell durch eine persönliche Webrecherche des Autors, wobei als Einstiegspunkt die Homepage der FSU Jena gewählt wurde (<http://www.uni-jena.de>). Ausgehend von den zehn Fakultäten der FSU Jena wurden die Institute und die Lehrstühle aufgenommen für die aussagekräftige Homepages existieren. Kleine Bereiche, die nur aus einer Webseite bestehen und dazu kaum Informationen enthalten, wurden zu Gunsten der Ontologiegöße und aufgrund ihrer geringen Relevanz für eine Websuche nicht in die Ontologie aufgenommen. Damit begründet sich auch die Tatsache, dass die in der Ontologie kodierte Wissensbasis *keinen* Anspruch auf Vollständigkeit erhebt.

Abbildung 14 zeigt eine Einschätzung über die Gesamtstruktur der „Webseitenlandschaft“ der FSU Jena, gruppiert in geeignete Themengebiete. Die angegebenen Prozentwerte sind dabei grobe Schätzungen des Autors und beruhen auf Erfahrungswerten während der Zeit der Initialerstellung der Ontologie. Die Grafik zeigt, welchen Anteil in etwa die einzelnen Gruppen an der Menge der über die Domäne der FSU Jena zugänglichen Webseiten haben. In der Ontologie haben hauptsächlich die *Fakultäten*, die *Institute* und ein großer Teil der *Lehrstühle* Eingang gefunden. Weiterhin wurden wichtige *Zentrale Einrichtungen*, *Verbände*, *Kliniken*, *Arbeitsgruppen*, etc. aufgenommen. Die Homepages der Mitarbeiter und Studenten wurden aufgrund der großen Anzahl und ihrer überwiegend geringen Bedeutung⁶⁹ nicht mit berücksichtigt. Auch die Hauptwebseiten der FSU Jena (*uni-jena.de*), die auf einem *Content Management System* (CMS) beruhen, konnten nicht berücksichtigt werden, da die angebotenen Webseiten völlig unstrukturiert hinter der Domäne *uni-jena.de* angeordnet sind und somit eine inhaltlichen Gruppierung themenverwandter Webseiten kaum möglich ist.

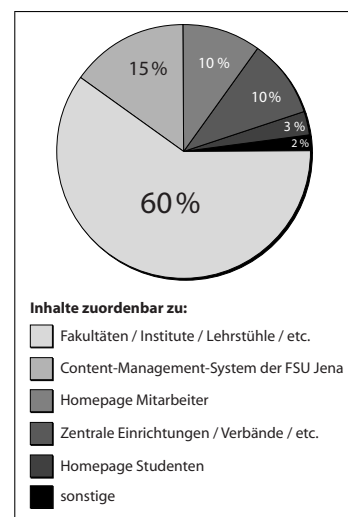


Abbildung 14: Themenklassen der Webseitenstruktur der FSU Jena.

Eine komplette Aufnahme aller Webseiten unter der Domäne *fsu-jena.de* wäre zwar wünschenswert, erwies sich aber als äußerst schwierig und hinderlich für die weitere Arbeit. Zum einen existiert für jede Struktureinheit nicht zwangsläufig eine eigene Homepage und zum anderen ist es nicht Ziel der Arbeit die Gesamtheit aller existierenden Webseiten zu sichten und zu analysieren. Vielmehr beruht der Satz der aufgenommenen Homepages der subjektiven Einschätzung des Autors. Bei genauerer Betrachtung ist darin sogar eine Stärke der Idee hinter dem *Sonto*^x-System zu sehen. Die Ontologie kann so, durch Aufnahme von nützlich und wichtig erscheinenden Konzepten, optimal angepasst werden.

Da ein semantisches Netz noch nicht existiert und die Angabe von geeigneten Metadaten von kaum einem Webmaster der einzelnen Bereiche auch nur ansatzweise umgesetzt wurde, war eine automatische Auswertung und Relevanzbewertung zum gegenwärtigen Zeitpunkt noch nicht möglich.

```

1  (...)
2  <Lehrstuhl rdf:ID="Lehrstuhl_für_Entwicklungspsychologie">
3    <Telefon rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
4      +03641 945200</Telefon>
5    <Anschrift rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
6      Am Steiger 3/1, D-07743 Jena</Anschrift>
7    <E-Mail rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
8      Rainer.Silbereisen@uni-jena.de</E-Mail>
9    <Picture rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
10     http://www.artusweb.de/SontoX/ontology/img/
11     Lehrstuhl_fuer_Entwicklungspsychologie.png</Picture>
12    <Homepage rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
13     http://www.uni-jena.de/svw/devpsy/</Homepage>
14    <Homepage rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">

```

⁶⁹ Bezogen auf die Aufgabenstellung

```
14   http://www2.uni-jena.de/svw/devpsy</Homepage>
15   <Inhaber rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
16     Rainer K. Silbereisen, Dr.phil.</Inhaber>
17   <gehört_zu rdf:resource="#Institut_für_Psychologie"/>
18   </Lehrstuhl>
19   (...)
```

Listing 18: Auszug aus der Individuen-Definition (fsu-jena.owl).

In Listing 18 wird am Beispiel des *Lehrstuhles für Entwicklungspsychologie* die durch Protégé generierte entsprechende XML-Kodierung einer Individuen-Definition in der Ontologie gezeigt. Das öffnende *Lehrstuhl*-Tag (Zeile 2) macht deutlich, dass es sich um eine Klasseninstanz der zuvor definierten *Lehrstuhl*-Klasse handelt. Der Wert für das Attribut *rdf:ID* stellt den Bezeichner der Instanz dar.

Abschließend zum Thema Wissensakquise soll darauf hingewiesen werden, dass sich der benötigte Aufwand für die Aufnahme aller Daten als unerwartet hoch erwies und daher nicht unterschätzt werden darf.

8.4 Ontologiepflege

Um zu garantieren, dass *Sonto*^x nur korrekte und aktuelle Zuordnungen und Daten anzeigt, ist eine gewisse Aktualität der Ontologie notwendig. Es liegt in der Natur des WWW, dass Webseiten bzw. ihre Strukturen sich schnell ändern. Um diese sich verändernden Gegebenheiten möglichst rasch erfassen zu können, muss die Ontologie in gewissen Zeitabständen auf Korrektheit und Konsistenz hin überprüft und ggf. angepasst werden. Nach Einschätzung des Autors sollte ein Überprüfungsintervall von bis zu drei Monaten angestrebt werden. In erster Linie ist die Aufgabe der Ontologiepflege,

- ❑ die Erreichbarkeit der URLs für jedes Individuum und
- ❑ die Korrektheit der Eigenschaftswerte der Individuen zu kontrollieren.

Darüber hinaus ist eine Kontrolle der Klassen-, und Eigenschaftsdefinitionen in einem größeren Zeitintervall angebracht, z. B. innerhalb von sechs Monaten. Hinsichtlich der Gültigkeit der URLs ist hier eine softwaregestützte Überprüfung denkbar, wodurch Probleme schnell und einfach erkannt werden können. Die Überprüfung der Definitionen muss manuell erledigt werden, jedoch wird der damit verbundene Aufwand als gering eingeschätzt.

Unter Nutzung von Protégé, ist es leicht möglich, die Ontologie einzulesen und dann im Anschluss die gewünschten Änderungen vorzunehmen.

8.5 Anforderungen an eine *Sonto*^x-konforme Ontologie

Obwohl in *Sonto*^x eine Datenunabhängigkeit bezüglich der verwendeten Ontologie angestrebt wurde, konnte dieser Anspruch nicht vollständig aufrechterhalten werden. Folgende vier Beschränkungen haben sich in der Umsetzungsphase des Systems ergeben und sind

bei einer Ontologieerstellung zu beachten, damit die Ontologie später in *Sonto*^x eingebunden werden kann:

- ① Verwendung von *Protégé* in der Version 3.0 als Garantie für eine korrekte umgesetzte OWL-Syntax sowie einer erfolgreichen Verarbeitung.
- ② Anlegen einer *Wurzel-Klasse*. Sie fasst alle anderen Klassen zusammen und stellt das oberste Element einer Organisationsstruktur dar. (Der Name der Klasse kann frei gewählt werden.)
- ③ Anlegen einer ObjectProperty *gehört_zu*. Für jedes neu aufgenommene Individuum muss als Wert für diese Eigenschaft ein übergeordnetes Individuum zugeordnet werden. Die Ausnahme stellt hier das Wurzel-Individuum dar.
- ④ Anlegen einer DatatypProperty *Homepage*. Eine Wertezuordnung für ein Individuum ist zwar nicht zwingend notwendig, wird jedoch dringend empfohlen, da für *Sonto*^x dieser Wert eine zentrale Rolle spielt. Ist kein URL für diese Eigenschaft vorhanden, ist zu überlegen, ob das entsprechende Individuum aus der Ontologie ganz entfernt werden kann.

Werden alle oberen Punkte beachtet, kann im Prinzip jede beliebige, auf diese Weise erstellte Ontologie einer Webseitenstruktur, in *Sonto*^x eingebunden werden.

9 Vorverarbeitung der Ontologie

Die erstellte Ontologie soll nun dafür verwendet werden, die Suchtreffer auf die enthaltenen Konzepte bestmöglich abzubilden und diese Zuordnung dem Nutzer später im Web-Interface kenntlich zu machen. Des Weiteren soll der hierarchische Zusammenhang der einzelnen Individuen geeignet repräsentiert werden. Hierzu ist es notwendig den Inhalt der Ontologie automatisch zu analysieren. Zur Lösung dieser Aufgabe wurde ein eigener OWL-Parser programmiert. Dieser ermöglicht die Abbildung des Inhaltes der OWL-Datei auf eine geeignete Datenstruktur zur Weiterverarbeitung.

Die Programmierung eines eigenen OWL-Parsers, war zu Beginn der Problembearbeitung nicht das angestrebte Ziel. Es wurde zuvor untersucht, ob für PHP eine entsprechende OWL-Parser-Klasse zur Verfügung steht. In der Tat findet sich hierfür ein viel versprechendes Projekt. Die Rede ist von der *RAP (RDF API für PHP)*, einem SW-Toolkit für PHP Entwickler.⁷⁰ Die Dokumentation [WB04] verspricht die Bereitstellung eines RDF/XML Parsers und, mit der integrierten *OntModel API*, eine Verarbeitungsmöglichkeit von Klassen, Eigenschaften und Individuen einer Ontologie.

Dem Autor ist es jedoch nicht gelungen die erstellte OWL-Datei in RAP einzulesen. Probleme bereitete hier die Integration des korrekten OWL-Namensraums. Nach genauerem Studium der Dokumentation fand sich ein Hinweis darauf, dass das Generieren eines Ontologie-Modells gegenwärtig nur unter Verwendung des RDFS-Vokabulars möglich ist. Eine Implementierung des OWL-Vokabulars sei für die Zukunft angedacht. Aus diesen

⁷⁰ RAP ist ein Open Source Projekt der *Freien Universität Berlin* und ist momentan unter <http://sourceforge.net/projects/rdfapi-php/> in der Version 0.9.1 zu beziehen.

Gründen war zum Entwurfszeitpunkt von *Sonto*^x der Einsatz von RAP nicht möglich. Für eine spätere Weiterentwicklung von *Sonto*^x könnte jedoch die Verwendung von RAP das Problem des OWL-Parsers effizient lösen und zusätzliche Funktionalitäten, wie z. B. die jetzt schon von RAP unterstützte *RDF Query Language (RDQL)* und ein Schlussfolgernssystem (*inference engine*) bereitstellen.

9.1 Entwicklung eines eigenen OWL-Parsers

Ein OWL-Parser soll die Analyse der in OWL kodierten Ontologie vornehmen und die Informationen dem *Sonto*^x-System in einer strukturierten Form bereitstellen. PHP stellt ab der Version 5 eine vereinfachte Methode zur XML-Verarbeitung zur Verfügung. Es handelt sich dabei um einen Satz an Funktionen, die einen effizienten Zugriff auf die Daten einer XML-Datei ermöglichen. Die zentrale Funktion ist dabei *simplexml_load_file()*, welche als Parameter den Pfad zu einer XML-Datei erwartet und diese einliest:

```
$this->xml = (object) simplexml_load_file((string) $owl_file);
```

Nach Abarbeitung obiger Programmzeile ist mittels einfacher Methoden⁷¹ ein objektorientierter Zugriff auf die Elemente der XML-Datei über die Objekt-Variable *\$this->xml* möglich. Es wird hierdurch nicht nur ein Zugriff auf die Daten in einem geklammerten Elemente-Tag, sondern auch auf die einzelnen Attribute eines Tags ermöglicht. Weiterhin wird zusätzlich eine Unterstützung des Namensraums bereitgestellt.

Für eine komplette Analyse der Ontologie ist eine komplexe und umfangreiche Abarbeitung der Elementestruktur der XML-Datei erforderlich. Schwierigkeiten bereiteten hier vor allem die unterschiedlichen, in der Ontologie genutzten Namensräume, wie z. B. *rdf*, *rdfs* und *owl*. Für jeden Zugriff auf ein Element bzw. dessen Attribute muss zusätzlich zum Namen der korrekte Namensraum angegeben werden. Die automatische Analyse der Ontologie wird dadurch erheblich komplexer und schwieriger als das Parsen einfacher XML-Dateien.

Das Vorhaben einer Implementierung eines universellen OWL-Parsers, der jede beliebige OWL-Ontologie verarbeiten kann, musste im Laufe der Arbeit jedoch fallen gelassen werden. Es zeigte sich, dass die Komplexität und der Arbeitsaufwand den Zeitrahmen dieser Diplomarbeit übersteigen. Obwohl OWL und vor allem OWL-Lite im Prinzip eine einfache Konzeptabbildung erlaubt, können damit ebenso sehr anspruchsvolle und umfangreiche Ontologien modelliert werden. Die Schwierigkeit bei der Programmierung eines Parsers lag hauptsächlich in der Mächtigkeit der OWL-Syntax, welche es erlaubt, ein und denselben Sachverhalt mit verschiedenen Sprachkonstrukten und an unterschiedlichen Stellen umzusetzen. Nicht zuletzt erhöht die Möglichkeit einer hohen Verschachtelungstiefe der OWL-Elemente die Komplexität einer automatischen Verarbeitung beträchtlich.

Der Anspruch einer möglichen Ontologieunabhängigkeit soll aber weitgehend aufrecht erhalten bleiben. Da für die Umsetzung der Universitätsontologie nur ein sehr kleiner Teil der möglichen Sprachelemente zum Einsatz gekommen ist, entstand die Idee einen

⁷¹ Z. B. kann mit der Funktion *children()* auf alle Sub-Elemente eines XML-Tags zugegriffen werden, während über *attributes()* ein Zugriff auf alle Attribute eines Tags möglich ist.

speziellen, eigens für die Gegebenheiten umgesetzten OWL-Parser, auf Basis einer OWL-Lite-Teilmenge, in Angriff zu nehmen. Die verwendeten Konstrukte erwiesen sich dabei als völlig ausreichend, um die für diese Aufgabenstellung benötigten Informationen zu modellieren.

9.2 Erstellung einer OWL Parser-Klasse

Die Funktionalität eines Parsers für die Analyse einer OWL-Datei wurde in einer eigenen Parser-Klasse (*OWLP.class*) umgesetzt. Die Klasse ermöglicht eine für diese Arbeit zugeschnittene Ontologieauswertung. Folgende Methoden wurden dafür programmiert:

- ❑ array **getClass**(object *SimpleXMLElement*)
- ❑ array **getObjectProperty**(object *SimpleXMLElement*)
- ❑ array **getDatatypeProperty**(object *SimpleXMLElement*)
- ❑ array **getIndividual**(object *SimpleXMLElement*, array *classes*)
- ❑ array **getIndividualRecursiv**(object *SimpleXMLElement*, string *classname*)

Bedeutend in *Sonto*^x ist die Methode *getIndividual()*, die die Hilfsmethode *getIndividualRecursiv()* aufruft. Die rekursive Methode übernimmt die eigentliche Arbeit, indem sie die Individuen-Definitionen unabhängig von der Verschachtelungstiefe der Elemente ausliest. Ein Objekt der OWLP-Klasse wird in der CONTROL-Klasse bei jeder gestellten Suchanfrage instanziiert:

```
$this->OWLP = (object) new OWLP((string) $owlOntologyFile);
```

Der Konstruktor der OWLP-Klasse parst bereits im Aufruf den Inhalt der übergebenen XML-Datei. Listing 19 zeigt den Aufruf der einzelnen Parser-Methoden im Konstruktor.

```

1  (...)
2  public function __construct($owl_file) {
3      $this->xml = (object) @simplexml_load_file($owl_file);
4      $this->classes = (array) $this->getClass((object) $this->xml);
5      // $this->getObjectProperty((object) $this->xml);
6      // $this->getDatatypeProperty((object) $this->xml);
7      $this->individuals = (array) $this->getIndividual((object) $this->
          xml, (array) $this->classes);
8  }
9  (...)
```

Listing 19: Konstruktor der OWLP-Klasse

Die richtige Reihenfolge der Methodenausführung ist hier zu beachten. So kann *getIndividual()* erst aufgerufen werden, wenn mit *getClass()* ein Array mit allen enthaltenen Klassen-Definitionen bereitgestellt wurde. Es zeigte sich, dass die Auswertung der Klassendefinition und der davon abgeleiteten Individuen, die benötigten Informationen für *Sonto*^x bereitstellen. Die zu Beginn mit umgesetzten Property-Methoden kommen daher

im momentanen *Sonto*^x-System nicht zum Einsatz, stehen jedoch für einen zukünftigen eventuellen Einsatz bereit.⁷²

Der Kern der Anwendung stellt das Array mit den enthaltenen Individuen dar, auf dessen Basis die gesamte Weiterverarbeitung beruht. Nach dem Parsen der Ontologie steht der CONTROL-Klasse ein komplexes Array mit allen Informationen der Individuen zur Verfügung. Listing 20 zeigt an einem gekürzten Auszug ein konkretes Beispiel für die Struktur des von der Methode *getIndividual()* zurückgegebenen Arrays. Dieses Array beinhaltet alle für *Sonto*^x benötigten Informationen aus der Ontologie und stellt die Basis für die weitere Verarbeitung dar.

```

1 Array (
2   (...)
3   [42] => Array (
4     [instanceOf] => Lehrstuhl
5     [ID] => Lehrstuhl_für_Entwicklungspsychologie
6     [subElements] => Array (
7       [0] => Array (
8         [element] => Inhaber
9         [value] => Rainer K. Silbereisen, Dr.phil. )
10      (...)
11      [7] => Array (
12        [element] => gehoert_zu
13        [value] => Institut_für_Psychologie )
14      [8] => Array (
15        [element] => Homepage
16        [value] => http://www.uni-jena.de/svw/devpsy/ ) ) )
17   (...)
18 )

```

Listing 20: Struktur des Individuum-Arrays.

9.3 Speicherung des Parserergebnisses

Es stellt sich die Frage, *ob* und *wie* das Ergebnis der Ontologietheorieanalyse abgespeichert werden soll, um im Weiteren einen effektiven Zugriff auf diese Informationen zu gewährleisten. Folgende Alternativen wurden zu Beginn in Betracht gezogen:

- ① Einmalige Parserausführung und Abspeicherung des Ergebnisses im Sekundärspeicher, um zur Laufzeit diesen Aufwand einzusparen.
- ② Parsen bei jeder ausgelösten Anfrage und Bereithalten des Ergebnisses über den System-Hauptspeicher.

Die erste Möglichkeit suggeriert einen Performancegewinn. Werden die Daten in einer Datei abgespeichert, müssen sie jedoch bei jedem Programmaufruf aus der Datei wieder eingelesen werden und der anfängliche Performancegewinn wird dadurch wieder geschmälert. Die Speicherung in einer Datenbank würde hingegen einen schnellen Zugriff auf die

⁷² In Listing 19 (Zeilen 5 und 6) sind diese daher auskommentiert wurden.

Daten über ein Datenbank-Management-System garantieren. Verlockend ist besonders die Möglichkeit einer integrierten Abfragesteuerung mittels der *Structured Query Language (SQL)*.

Für diese Arbeit wurde sich jedoch gegen diese Möglichkeit entschieden, da der Aufwand für den Einsatz eines speziellen Datenhaltungssystems in keiner vernünftigen Relation zu der tatsächlich anfallenden Datenmenge steht.

Obwohl die zweite Möglichkeit auf den ersten Blick die weniger schöne Variante der beiden darstellt, zeigt sich, dass der zu vermutende Performanceverlust relativ gering ausfällt. Tabelle 3 zeigt einen Performancevergleich⁷³ des Parsers für unterschiedliche Dateigrößen der Ontologie.

Tabelle 3: Parser-Ausführungszeit.

Dateigröße der Ontologie	Ausführungszeit des Parsers	
	gesamt	<i>simplexml_load_file()</i>
100 Kilobyte	0.03 Sekunden	0.005 Sekunden
500 Kilobyte	0.23 Sekunden	0.04 Sekunden
1000 Kilobyte	0.74 Sekunden	0.08 Sekunden

Unbestritten würde bei einer Dateigröße über 1 MB eine spürbare Verzögerung deutlich. Die wahre Größe der vorliegenden Ontologie liegt jedoch bei ca. 100 KB und wird selbst bei einer späteren eventuellen Erweiterung die Größe von 150 KB kaum überschreiten. Die Methode des erneuten Parsens der Ontologie für jeden einzelnen Programmaufruf wird daher als tolerierbar für das Gesamtsystem angesehen und in *Sonto*^x angewendet.

10 Verbindung der Ontologie mit der Datengrundlage einer Suchmaschine

Die Programmkomponente, die für die Steuerung der Logik des Web-Interfaces zuständig ist, übernimmt zugleich die Integration der Ontologie und die Verwaltung der Suchmaschinentreffer. Diese Kernfunktionen wurden in der zuvor angesprochenen CONTROL-Klasse umgesetzt. Die Klasse selbst stellt die Hauptklasse von *Sonto*^x dar, in der alle anderen Klassen der Klassenbibliothek eingebunden werden. Im Web-Interface werden konsequent nur Methoden der CONTROL-Klasse aufgerufen. Die CONTROL-Klasse stellt, auf der Ebene der Klassen eine Schnittstelle zwischen dem Web-Interface und den beiden Komponenten Suchmaschinentreffer und Ontologie dar.⁷⁴

⁷³ Die gemessenen Zeiten stellen den Mittelwert für jeweils zehn Parser-Aufrufen pro Dateigröße dar.

⁷⁴ Siehe System-Architektur in Anhang B auf Seite 79.

An dieser Stelle ist es angebracht, einen Blick auf die Systemkomponenten anhand der Verzeichnis- und Dateistruktur zu werfen. Dadurch soll ein besseres Verständnis der noch vorzustellenden Techniken, im Hinblick auf den Ort ihrer Umsetzungen, erreicht werden. Abbildung 15 zeigt die Verzeichnis- und Dateistruktur des *Sonto^X*-Systems, wie sie in der fertigen Anwendung auf dem Web-Server vorzufinden ist.

Das Verzeichnis *lib* stellt die Klassenbibliothek dar und beinhaltet alle sechs PHP-Klassendefinitionen die in *Sonto^X* verwendet werden. Welche Klasse dabei welche Funktionalitäten bereithält, wird in den kommenden Abschnitten näher erläutert.

Alle für die Ontologie relevanten Dateien sind unter dem Verzeichnis *ontology* angeordnet. Hierzu zählt außer der Ontologie (*fsu-jena.owl*) selbst, eine mit Protégé erzeugte HTML-Präsentation⁷⁵ (Unterverzeichnis *html*) und die momentan an dieser Stelle untergebrachten Bilder, die einigen Individuen zugeordnet wurden (Unterverzeichnis *img*).

Die Datei *search.php5* stellt das Zentrum der Webanwendung dar, in der letztendlich alle Fäden zusammenlaufen und mit welcher der Nutzer per Browser interagiert. Weiterhin existiert als Einstiegsseite die Datei *index.html* und als Optionsseite die Datei *adv_search.php5*. Dieses klassische Dreigespann ist in den Web-Schnittstellen der meisten Suchmaschinen anzutreffen und wurde auch für *Sonto^X* übernommen, da sich diese Struktur bewährt hat und die meisten Nutzer damit vertraut sind. Auf eine genaue Erläuterung der klassischen Benutzerführung einer Suchmaschine wird daher hier verzichtet. In Anhang C auf Seite 80 sind jeweils Screenshots aller drei Webseiten des Web-Interfaces abgebildet. Die beiden Webseiten *analyse.php5* und *info.php5* werden vom Nutzer nicht direkt aufgerufen, sondern kommen vielmehr als Darstellungshilfe in der Datei *search.php5* bei Bedarf zum Einsatz. Die Webseite *about.html* gibt dem Nutzer Hilfestellung bei der Benutzung der *Sonto^X*-Suche und Interpretation der Ergebnisse.

Speziell zur Realisierung des Web-Interfaces, im Hinblick auf das Webdesign, wird die Datei *style.css* (Stylesheets zur Formatierung der Ausgabe) und das Verzeichnis *img* (Grafiken des Web-Interfaces) benötigt. Die Datei *config.php* beinhaltet einige globale Konfigurationseinstellungen des Systems.

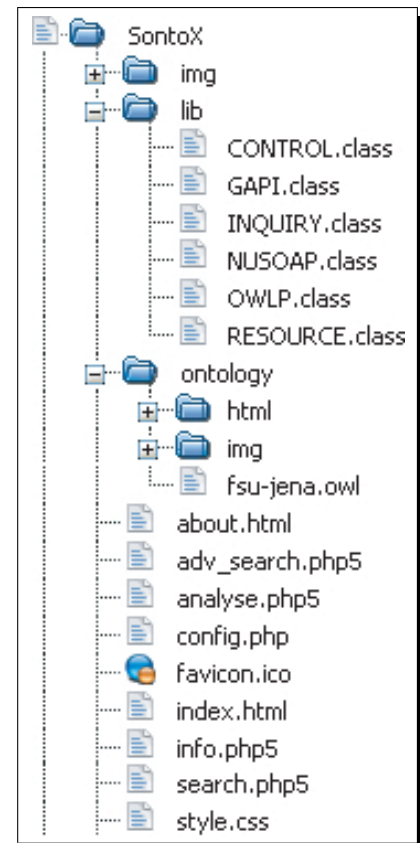


Abbildung 15: Verzeichnis- und Dateistruktur des *Sonto^X*-Systems.

⁷⁵ Abrufbar unter <http://www.artusweb.de/SontoX/ontology/html/index.html>

Tabelle 4: Implementierte Methoden der CONTROL-Klasse

Methode	Beschreibung / Funktionalität
<i>__construct()</i> *	Konstruktor der CONTROL-Klasse
<i>parseOntology()</i> *	Partern der Ontologie zur Informationsgewinnung
<i>startPageCheck()</i>	Überprüft, ob die Anfrage von der Startseite gestellt wurde
<i>initINQUIRY()</i> *	Auslösen einer Anfrage an eine Suchmaschine
<i>getURLforResource()</i>	Ermittelt alle URLs für ein Individuum der Ontologie
<i>mapHomepage()</i> *	Versuch der Zuordnung eines URL zu einem Individuen-spezifischen URL
<i>get_status()</i> *	Generiert die Statuszeile oberhalb der Trefferliste
<i>get_navigation()</i> *	Generiert die Seitennavigation unterhalb der Trefferliste
<i>getNamefromURL()</i>	Gibt den zugeordneten Namen (ID) eines URL zurück
<i>getURLfromName()</i>	Gibt den zugeordneten URL (Homepage) eines Namen zurück
<i>makeSubTaxo()</i>	Generiert die Sub-Taxonomie unterhalb der aktuellen Ebene der Suchraumeinschränkung
<i>getRootElement()</i>	Rekursive Methode um für eine beliebige Ressource das Wurzel-Element in der Ontologie zu finden
<i>fuzzy_control()</i>	Ist für die Steuerung der unscharfen Suche (Fuzzy-Suche) zuständig
<i>getTaxo()</i> *	Generiert die Taxonomie der Webseiteneinschränkung
<i>getInfo()</i> *	Ist für die Informationsaufbereitung einer Webressource in einem Inline-Frame zuständig
<i>makeParamterString()</i>	Generiert den Parameterstring für den Skriptaufruf
<i>makeParaStrforTaxo()</i>	Generiert den Parameterstring für den Skriptaufruf speziell für Links der Taxonomie
<i>getResults()</i> *	Gibt die Treffer der Suchabfrage in formatierter Form zurück
<i>getAnalyseInfo()</i> *	Versucht die Meta-Daten einer Webressource auszulesen, falls <i>mapHomepage()</i> erfolglos war

Alle in der CONTROL-Klasse implementierten Methoden sind in der Tabelle 4 zusammenfassend mit einer jeweiligen kurzen Beschreibung aufgelistet. Die mit einem Stern (*) markierten Methodennamen stellen dabei die Hauptfunktionalitäten bereit, während die unmarkierten Methoden als Hilfsmethoden in den Hauptmethoden zum Einsatz kommen. Die wichtigsten Methoden der Tabelle 4 werden an späterer Stelle anhand eines jeweiligen Beispiels genauer vorgestellt.

Da die Implementierung untrennbar mit der visuellen Aufbereitung des Web-Interfaces verbunden ist, werden die umgesetzten Techniken und die dafür eingesetzten Methoden in engem Zusammenhang mit der endgültigen dem Nutzer dargebotenen Form erläutert. Zu diesem Zweck werden die umgesetzten Ideen anhand eines jeweiligen Ausschnittes des Web-Interfaces vorgestellt und deren Umsetzung anschließend genauer besprochen. Auf das Thema Webprogrammierung in Hinblick auf das Web-Design, HTML, XHTML,

CSS, Browserkompatibilität, Variablenübergabe, Benutzerführung, etc. wird in dieser Arbeit nicht eingegangen, da dies eher eine „handwerkliche“ Komponente darstellt, die zwar sehr wichtig für das Gelingen einer Web-Anwendung ist, jedoch in dieser Arbeit vorausgesetzt wird.

10.1 Zuordnung der Suchtreffer zu den Individuen

Zuerst wurde sich darüber Gedanken gemacht, wie eine Zuordnung der gelieferten Suchtreffer zu den in der Ontologie aufgenommenen Individuen möglich ist und wie diese kenntlich gemacht werden kann. In Abbildung 16 ist die Darstellung einer gelungenen Zuordnung zu sehen.

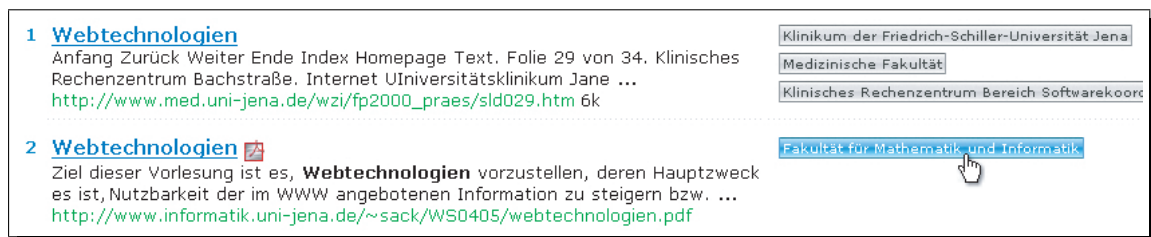


Abbildung 16: Beispielzuordnung der Individuen zu den Treffern.

Im Beispiel (Abb. 16) wurde nach dem Begriff „Webtechnologien“ gesucht. Auf der linken Seite sind die ersten beiden Treffer, der von der Google API gelieferten Trefferliste zu sehen. Auf der rechten Seite wird in *Sonto*^x eine Zuordnung der jeweiligen Webresource zu den in der Ontologie enthaltenen Individuen mittels Einblendung der jeweiligen Namen angezeigt.⁷⁶ Wie Treffer eins in Abbildung 16 zeigt, sind auch mehrere Zuordnungen möglich.

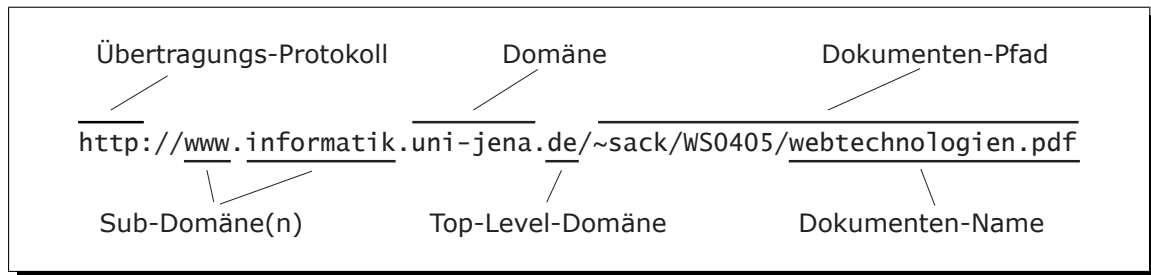
Mit einem Klick auf die jeweilige erkannte Strukturebene der FSU Jena, kann der Nutzer seine Suche auf die entsprechende zugeordnete Domäne einschränken. Im Beispiel würde ein Klick auf die Zuordnung *Fakultät für Mathematik und Informatik* die Suche auf die, der Fakultät für Mathematik und Informatik zugeordneten Domänen, für die Eigenschaft *Homepage* erfolgen. Wie dies im Detail umgesetzt wurde, wird im Abschnitt 10.2 erläutert.

Funktionsweise der Zuordnung

Der erste und zugleich nahe liegende Ansatz zum Information Retrieval besteht in der Auswertung der syntaktischen URL-Struktur einer Webseite. Die weltweit einmalige Adressierung liefert einen ersten Ansatzpunkt zur inhaltlichen Zuordnung der Webseite. Für eine exakte Auswertung des URL, muss ein Blick auf die formale Syntax geworfen werden.⁷⁷ Abbildung 17 zeigt für obiges Beispiel die einzelnen URL-Komponenten mit ihrer jeweiligen Bedeutung.

⁷⁶ Der Name entspricht dabei dem Attributwert von *rdf:ID* der Individuen-Definition in der Ontologie.

⁷⁷ Siehe Spezifikation in RFC 1738 und 1808.

**Abbildung 17:** URL-Struktur anhand eines Beispiels

In *Sonto*^X wird untersucht, ob ein Homepage-Wert aus der Ontologie, eine Teilmenge der URL eines Suchtreffers darstellt. Ist dies der Fall, wird das jeweilige Individuum der Webressource zugeordnet. Da dies für jede Webressource und über alle URLs der Ontologie erfolgt, sind auch Mehrfachzuordnungen möglich.

Die Funktionalität wird über die *mapHomepage()* Methode der CONTROL-Klasse bereitgestellt. Die Methode *mapHomepage()* erwartet als Argument einen URL und wird in der Methode *getResults()* für jeden Suchtreffer ausgeführt (Listing 21, Zeile 5).

```

1 public function getResults() {
2     (...)
3     foreach ($this->INQUIRY->Resources as $Resource) {
4         (...)
5         $xhtml = (string) $this->mapHomepage((string) $Resource->url);
6         (...)
7     }
8     (...)
9 }

```

Listing 21: Aufruf von *mapHomepage()* in *getResult()*

Der Rückgabewert der *getResults()*-Methode enthält den XHTML-Quelltext, der die Treffer und die eventuell zugeordneten semantischen Erweiterungen, in einer für den Browser aufbereiteten Form (Abb. 16, rechts) enthält.

10.2 Einschränkung des Suchraumes

Die meisten Suchmaschinen bieten die Möglichkeit, die Suche auf eine bestimmte Domäne zu beschränken. Auf einer Optionsseite muss dafür der entsprechende Domänen-Name in ein Formular eingetragen werden.⁷⁸

In *Sonto*^X wird der Nutzer von dieser Mühe befreit, da die Angabe der entsprechenden Domäne bei jeder ausgelösten Suchraumeinschränkung automatisch vorgenommen wird.

Der konkrete Mechanismus der Suchraumeinschränkung hängt jedoch von der eingesetzten Suchmaschine ab. Google gibt den Nutzer die Möglichkeit, über eine Suchstringerweiterung mittels dem Konstrukt *site:* die Suche einzuschränken. Dieser Mechanismus wird

⁷⁸ Alternativ kann diese Angabe auch in dem Suchstring mit untergebracht werden, wo sie im Endeffekt immer mit übergeben wird.

auch durch die Google API unterstützt. Als Parameter wird der Domänen-Name des Web-Servers übergeben, auf dessen Seitenbestand die Suche beschränkt werden soll. Zum Beispiel würde ein Klick auf die untere Zuordnung in Abbildung 16, eine neue Suchanfrage mit dem Suchstring „Webtechnologien site:minet.uni-jena.de“ auslösen. *Sonto*^X erweitert hierzu den eigentlichen Suchstring durch Angabe der jeweiligen Domäne.

Um unabhängig von der konkret eingesetzten Suchmaschine zu sein, wurde die Modellierungsfunktion in der INQUIRY-Klasse untergebracht. Die Suchstringerweiterung wird dort in der Methode *makeQueryString()* vorgenommen. In ihr wird momentan die Google spezifische *site:-*Angabe zusammengestellt.

Die Kontrolle der aktuellen Sucheinschränkung wird durch den Parameter *rn* (resource name) gesteuert, welcher den *rdf:ID*-Namen für die jeweilige Strukturebene enthält, so wie er in der Ontologie angegeben wurde. Der *Sonto*^X-Skriptaufruf

```
(...)search.php5?q=testquery&rn=Fakultät für Mathematik und  
Informatik,
```

führt zur Modellierung folgender Suchstringerweiterung:

```
testquery site:minet.uni-jena.de.
```

Unter *minet.uni-jena.de* sind die Webseiten der Fakultät für Mathematik und Informatik im WWW zu erreichen. Google wird durch diese Angabe veranlasst, nur solche Treffer zurückzugeben, die unter dieser Domäne angeordnet sind. An dieser Stelle tritt jedoch ein Problem zu Tage: So können für ein und denselben Web-Server unterschiedliche Domänen-Namen existieren. Diesem Problem widmet sich der nächste Abschnitt.

10.2.1 Multiple Webresource-URLs

Google erlaubt nach eigenen Angaben nur eine einmalige Verwendung des *site:-*Konstruktes pro Suchanfrage [Google]. Existieren für einen Web-Server unterschiedliche (multiple) Domänen-Namen, ist es im Prinzip nur möglich, einen dieser Namen auszuwählen und der Suchmaschine als Parameter zu übergeben. Dies führt jedoch dazu, dass all diejenigen Suchtreffer der alternativen Domänen-Namen nicht mit in der Trefferliste berücksichtigt werden. Es zeigte sich, dass dadurch ein großer Teil des Webangebotes der jeweiligen Webseiten ausgeblendet wurde. Leider existieren im WWW und auch auf den WWW-Servern der FSU Jena viele Homepages, die unter unterschiedlichen Domäne-Namen erreichbar sind.

Technisch gesehen handelt es sich meist dabei um zusätzliche Alternativ-Einträge (sog. Alias-Namen) im *Domain-Name-System* (DNS) für dieselbe physische Web-Server-Adresse. Ein Grund für diese Umsetzung kann eine gewachsene Domain-Namen-Struktur sein, die sich im Laufe der Zeit angepasst oder verändert hat. Ein konkretes Beispiel hierfür sind die Alias-Namen *minet.uni-jena* und *informatik.uni-jena*, welche im DNS für den gleichen Web-Server eingetragen sind, auf dem die Homepage der Fakultät für Mathematik und Informatik hinterlegt ist.

Um dieses Problem in den Griff zu bekommen, musste für *Sonto*^x eine geeignete Lösung gefunden und umgesetzt werden. In dem folgenden Unterabschnitt wird eine Teillösung des Problems näher erläutert.

10.2.2 Behandlung multipler URLs und der Verzeichnisstrukturen

Um trotz unterschiedlicher Domänen-Namen für denselben Bereich die volle von dem Google-Server bereitgehaltene Treffermenge zu erhalten, wurde versucht, das *site*:-Konstrukt mehrfach anzuwenden. Obwohl in [Google] darauf hinweisen wird, dass diese Angabe nur einmal pro Suchanfrage berücksichtigt wird, zeigte dieses Vorgehen den gewünschten Erfolg. Google berücksichtigte alle angegebenen Domänen-Namen und erlaubt so einen Zugriff auf den gesamten indizierten Seitenbestand zu einem bestimmten Web-Server-Bereich. Bedingung für eine korrekte Funktionsweise ist jedoch die Angabe des booleschen Operators *OR* zwischen den jeweiligen *site*:-Konstrukten.⁷⁹

Um mehrere URLs berücksichtigen zu können, wurde bereits in der Ontologiedefinition selbst die Möglichkeit geschaffen, mehrere URLs für ein Individuum anzugeben. Welche alternativen URLs für ein Individuum vorliegen, wird mit Hilfe des vom OWL-Parser bereitgestellten Individuen-Array in der *CONTROL*-Klasse ermittelt. Der *INQUIRY*-Klasse wird bei der Instanzierung dann ein Array der verschiedenen URLs übergeben. In der Methode *makeQueryString()* wird mit Hilfe dieses Arrays der Suchstring mit den entsprechenden *site*:-Angaben mittels *OR*-Verknüpfung erweitert. Für den Suchbegriff „Webtechnologien“ mit der aktuellen Suchraumeinschränkung für die *Fakultät für Mathematik und Informatik* hat der endgültige Suchstring folgenden Inhalt:⁸⁰

Webtechnologien site:minet.uni-jena.de OR site:informatik.uni-jena.de

10.2.3 Zusätzliches Problem mit den Verzeichnisstrukturen

Ein weiteres Problem tritt bei unterschiedlichen Verzeichnisstrukturen (und gleichzeitigen multiplen URLs) eines Individuums auf. Im Folgenden soll anhand zweier Problemfälle beispielhaft deren prinzipielle Bearbeitung skizziert werden.

1. Fall: Unterschiedliche URLs mit identischen Pfad (z. B. *Institut für Informatik*)

- ❑ <http://www.minet.uni-jena.de/www/fakultaet>
- ❑ <http://www.informatik.uni-jena.de/www/fakultaet>

Für die Homepage des *Institutes für Informatik* ist zusätzlich die Pfadangabe */www/fakultaet* von Bedeutung. Google erlaubt jedoch nur die Sucheinschränkung über Domänen. In *Sonto*^x wird in diesem Fall der von Nutzer eingegebene Suchbegriff mit der zusätzlichen Angabe des Pfades erweitert. Die unterschiedlichen Domänen werden wie oben beschrieben mit dem *site*:-Konstrukt an den Suchstring angehängen. Google verknüpft

⁷⁹ Der boolesche Operator *AND* wird implizit durch die Angabe eines Leerzeichens ausgedrückt.

⁸⁰ In Wirklichkeit wird hier noch die Domäne *mathematik.uni-jena.de* angegeben, die einen weiteren DNS-Aliasnamen der Fakultät darstellt.

den Suchbegriff und die Pfadangabe durch ein logisches *AND*. Dies bewirkt, dass nur solche Webdokumente in der Trefferliste erscheinen, die zusätzlich zu dem Suchbegriff die Zeichenkette „/www/fakultaet“ enthalten. Da Google auch die URL zur Indizierung der Webseiten nutzt, werden nur Webseiten des Instituts für Informatik zurückgegeben, da nur für diese der richtige Pfad in der URL enthalten ist. Prinzipiell ist aber nicht auszuschließen, dass in obigen Fall auch Treffer anderer Bereiche im Ergebnis gelistet werden. Jedoch zeigte sich, dass dies bei Google nicht der Fall war.

Für den 1. Fall würde in *Sonto*^x folgender Suchstring an Google übergeben:

Webtechnologien /www/fakultaet site:minet.uni-jena.de OR site:informatik.uni-jena.de

2. Fall: Unterschiedliche URLs mit verschiedenen Pfaden (z. B. *Biologisch Pharmazeutischen Fakultät*)

☐ <http://pinguin.biologie.uni-jena.de/fakultaet/>

☐ <http://www2.uni-jena.de/biologie/>

Für die *Biologisch Pharmazeutischen Fakultät* treten zu den unterschiedlichen Domänen, zusätzliche sich unterscheidende Pfadangaben in Erscheinung. Hierbei handelt es sich um ein Problem, dass in *Sonto*^x nicht 100 %ig behandelt werden konnte und nur durch einen Kompromiss teilweise gelöst ist.

Des Weiteren stellt dies einen besonders problematischen Fall dar: Die beiden Domänen-Namen sind keine DNS-Alias-Namen für den gleichen Webserver, sondern stehen für zwei unterschiedliche IP-Adressen. Welcher Sinn hinter dieser augenscheinlichen Spiegelung des Webangebotes auf zwei Web-Server steckt, bleibt den Autor verborgen.

Werden beide Pfade wie zuvor beschrieben, an den Suchbegriff per *AND*-Verknüpfung angehängen, liefert Google keine Treffer. Der Grund liegt darin, dass nur solche Webseiten der angegebenen Domänen berücksichtigt werden, bei den zu dem Suchbegriff die Zeichenfolge /*fakultaet*/ und /*biologie*/ enthalten ist. Da dies entweder in dem ersten oder zweiten URL der Fall ist, aber nicht bei beiden gleichzeitig, kann Google keine Übereinstimmung mit seinem indizierten Seitenbestand finden.

In *Sonto*^x wird in solch einem Fall eine von den beiden Pfadangaben zufällig ausgewählt und an den Suchstring angehängen. Für den 2. Fall wählt *Sonto*^x eines der beiden folgenden Möglichkeiten aus:

Webtechnologien /fakultaet/ site:pinguin.biologie.uni-jena.de OR site:www2.uni-jena.de
oder

Webtechnologien /biologie/ site:pinguin.biologie.uni-jena.de OR site:www2.uni-jena.de

Die Methode der automatischen Suchstringerweiterung mit der Pfadangabe, kommt in *Sonto*^x auch zum Einsatz, wenn für ein Individuum nur eine einzige URL existiert, diese aber eine Pfadangabe enthält. Dann muss zur korrekten Suchraumeinschränkung dieser Pfad auch zusätzlich zu dem Suchbegriff angehängen werden, da auch hier die Angabe des alleinigen Domäne-Name nicht die richtigen Treffer liefern würde.

10.2.4 Probleme mit der URL-Struktur

Als äußerst ungeeignet zeigten sich die momentane URL-Struktur der WWW-Seiten der FSU Jena unter der Domäne *uni-jena.de*. Die Verzeichnis- bzw. Seitennamen stellen sich dort z. B. oft als unlesbar heraus.⁸¹ Grund dafür ist das eingesetzte Content-Management-Systems (CMS), welches zur internen Organisation der Webseiten nicht einen eindeutig lesbaren Namen nutzt, sondern auf eine Nummerierung setzt. Nach der Einschätzung des Autors ist dies auf eine konzeptuelle Designschwäche oder auf einer mangelnden Aufmerksamkeit zurückzuführen. Die Universität Jena steht mit diesem Problem leider nicht alleine da. Vielmehr finden sich solch „kryptische“ URLs bei vielen Webangeboten, welche zu- meist wohl ebenfalls auf die Verwendung eines CMS zurückzuführen sind. Als Beispiel ist hier die Homepage der Fachhochschule Jena (***www.fh-jena.de***) zu nennen, die ebenfalls auf ein CMS setzt, welches eine völlig Nutzer- und Suchmaschinen unfreundliche URL-Struktur verwendet. Da ein CMS zumeist von Fachleuten programmiert wird und finanziell eine erhebliche Investition bedeutet, ist es dem Autor unverständlich, dass diese Designschwächen auftreten und dass die Organisationen das Problem nicht erkennen.

Abschließend bleibt zu diesem Thema Suchraumeinschränkung festzuhalten, dass die Probleme durch unterschiedlichen URLs für dieselbe Homepage nicht vollständig gelöst werden konnten. Vor allem unterschiedliche Domänen mit unterschiedlichen Pfadangaben zu einer Homepage, stellen ein großes Problem dar. Da dies jedoch auf fragwürdige Webseitenstrukturen der jeweiligen Bereiche beruht, soll die daraus entstehende Ungenauigkeit bei der *Sonto*^x-Suche nicht dem System selbst zu Lasten gelegt werden. Eine saubere und dem URI-Konzept entsprechende Webseitenstruktur würde diese Probleme im Ansatz vermeiden. Vielleicht gibt diese Arbeit an richtiger Stelle ein Anstoß für ein Umdenken der verantwortlichen Webmaster.

10.3 Erweitertes Information Retrieval

In *Sonto*^x kommt bei Bedarf ein erweiterter Information Retrieval-Ansatz zum Einsatz. Schlägt eine Zuordnung wie in Abschnitt 10.1 beschrieben fehl, werden die Suchtreffer analog zur Google-Trefferliste ohne weitere Semantik bezüglich der Ontologie angezeigt. *Sonto*^x sollte hierfür aber eine alternative Lösung bieten. Es wurde untersucht, welcher alternative IR-Ansatz adäquate Informationen über die jeweilige Web-Ressource extrahieren kann. Dieser Ansatz soll dann alternativ ausgeführt werden, wenn die exakte Zuordnung auf Basis der URL keinen Erfolg hatte.

Die erste Idee bestand darin, während der Programmlaufzeit jede einzelne Webressource der Trefferdekade zu parsen⁸². Für diesen Zweck wurde ein eigener HTML-Parser

⁸¹ Z. B. ***www.uni-jena.de/content_page_0815.html***

⁸² Parsen bedeutet in diesem Fall das Analysieren der in HTML oder XHTML geschriebenen Webdokumente auf ihren Inhalt.

geschrieben, der pro Webseite als Information zusätzlich den enthaltenen Text und die wichtigsten Meta-Daten zur Verfügung stellt.⁸³

Es zeigte sich, dass dieses Vorgehen in einer „Sackgasse“ mündete. Es treten hier die gleichen Probleme auf, mit denen die Suchmaschinen zu kämpfen haben. Zum einen zeigte sich, dass eine Zuordnung einer Webseite auf Basis der Schlüsselwörter zu einem Individuum unmöglich war. Die fehlende Semantik und die teils mangelhafte HTML-Programmierung ließen eine Zuordnung nicht zu. Zum anderen nahm das Parsen der einzelnen Webseiten zu viel Zeit in Anspruch, sodass sich die Gesamtlauzeit der Programmausführung unakzeptabel in die Länge zog. Weiterhin verfolgt dieser Ansatz genau das Vorgehen der Suchmaschinen. Das wirft die Frage auf, warum nicht gleich eine eigene Suchmaschinen-Implementierung angegangen wird. Dies wäre die bessere Lösung, ist aber nicht das erklärte Ziel dieser Arbeit. Aus diesen Gründen wurde die Idee fallen gelassen und ein alternativer praxistauglicher Ansatz gesucht.

Eine automatische Bestimmung der Semantik anhand des Webseitentextes führte nicht zum Ziel, jedoch enthalten die Webseiten, wenn auch rudimentär, semantische Informationen in einigen Meta-Tags. Zumindest diese sollten für *Sonto*^x ausgelesen und aufbereitet werden. Abbildung 18 zeigt als Beispiel die ausgelesenen Annotationen der Meta-Tags einer Webseite so wie sie in *Sonto*^x umgesetzt wurde.



Abbildung 18: Alternativer IR-Ansatz

In Abbildung 18 (rechts) sind Angaben zum Autor (*author*), zur Beschreibung (*description*) und zu den Schlüsselwörter (*keywords*) einer Webseite zu sehen. Das Beispiel zeigt den Idealfall einer annotierten Webseite, wie er nur in Ausnahmefällen anzutreffen ist. Der weitaus überwiegende Teil der Webautoren macht kaum Gebrauch von dieser einfachen Art der semantischen Auszeichnung. Liegen Annotationen vor, dann sind die Inhalte oftmals kaum brauchbar oder einfach fehl am Platz. Nichtsdestotrotz wurde diese Funktionalität in *Sonto*^x aufgenommen, da fehlerhafte oder fehlende Annotation keine Systemchwäche von *Sonto*^x darstellen, sondern in den Verantwortungsbereich der jeweiligen Webautoren fallen.

Umsetzung der alternativen Meta-Tag-Analyse

Um diese Informationen zu erhalten ist ein Parsen der Webseiten nötig, wobei obiges Performance-Problem erneut zu Tage tritt. Passenderweise existiert unter PHP die Funktion *getMetaTags()*, die die Meta-Tags einer Webseite ausliest und den Inhalt in einem

⁸³ Es sei hier auf den Abschnitt Suchmaschinen im Teil I dieser Arbeit verwiesen, bei dem die Schwierigkeiten der automatischen Analyse von Webseiten beschrieben wurden. Das Gleiche trifft auch hier zu.

assoziativen Array bereitstellt. Da diese Funktion einzig den Header jeder Webseite auslesen muss, erwies diese sich als relativ schnell. Eine gewisse Verzögerung, wenn auch eine recht geringe, tritt dabei jedoch trotzdem auf. Weiterhin kann es vorkommen, dass lange Antwortzeiten der jeweiligen Web-Server dazu führen, dass die gesamte *Sonto^X*-Ergebnisseite sich mit einer großen Verzögerung im Browser des Nutzers aufbaut, da die Webseite erst vollständig vom Web-Server generiert werden muss, bevor sie übertragen wird. Zur Lösung dieses Problems werden die Meta-Tag-Informationen in einem, in der *search.php5*-Webseite eingebetteten Frame angezeigt.⁸⁴ Dies hat den Vorteil, dass die Inhalte der Hauptseite von *Sonto^X* komplett im Browser des Benutzers erscheinen, während in den jeweiligen eingebetteten Frames die Abfrage der Meta-Tags noch im Gange ist. Das Ergebnis ist eine Gesamtpräsentation, die nicht einen störenden Eindruck einer Anfrageverzögerung hinterlässt.

Die Datei *analyse.php5*, aus dem Stammverzeichnis von *Sonto^X*, wird dazu mit dem jeweiligen URL-Parameter in den eingebetteten Frame geladen. Die PHP-Funktion *getMetaTags()* ist in dieser Datei untergebracht und stellt für jedes Frame eine Anfrage. Wenn keines der drei ausgewählten Meta-Tags vorhanden ist, bleibt der Frame leer.

In der schon angesprochenen Methode *getResults()* wird immer dann, wenn eine Zuordnung fehlschlägt, die Methode *getAnalyseInfo()* aufgerufen:

```
$xhtml = '<td>'. $this->getAnalyseInfo($Resource->url). '</td>';
```

Der Rückgabewert ist vom Typ *string* und enthält den modellierten XHTML-Quelltext der in der Abbildung 18 (rechts) angezeigten Meta-Tag-Auswertung.

Obwohl dieser Ansatz allein auf Basis der drei Meta-Tags beruht, zeigte sich im Praxisinsatz von *Sonto^X*, dass bei entsprechenden aussagekräftigen Inhalten der Meta-Tags, dem Nutzer auf einem Blick wichtige Informationen über den Inhalt der jeweiligen Webresource zur Verfügung stehen, ohne diese besuchen zu müssen. Es soll hier nochmals darauf hingewiesen werden, dass der Nutzen dieses Ansatzes von der Qualität der von den jeweiligen Webmastern gemachten Annotationen abhängig ist.

10.4 Hilfestellung auf Basis der Ontologie für die Suchraumeinschränkung

Dem Nutzer soll über die Zuordnung der Treffer zu den in der Ontologie enthaltenen Konzepten hinaus eine Hilfestellung für eine individuelle Suchraumeinschränkung gegeben werden. Die in der Ontologie enthaltenen Informationen bildet dafür die Grundlage. Angestrebt wurde eine Visualisierung der Individuen, die über die Eigenschaft *gehört_zu* in der Ontologie miteinander in Beziehung stehen. Die so angezeigte Taxonomie, soll den Nutzer über die momentan gewählte Suchraumeinschränkung informieren und ihn zusätzlich die Möglichkeit einer alternativen Suchraumeinschränkung geben.

⁸⁴ Eingebettete Frames (*iframe* = *inline frame*) sind Bestandteil des XHTML-Standards und erzeugen in einer Webseite einen eigenständigen Bereich, in dem es ermöglicht wird Inhalte einer anderen Webseite anzuzeigen.

Abbildung 19 zeigt eine Beispielausprägung der in *Sonto*⁸⁵ generierten Taxonomie.⁸⁵ Die im Beispiel aktuelle Ebene ist das *Institut für Informatik*, welche als Zeichen der momentanen Suchraumeinschränkung blau hinterlegt ist. Das heißt, es werden nur Treffer angezeigt, die unter der Homepage der *Institutes für Informatik*⁸⁶ angeordnet sind.

Oberhalb der aktuell gewählten Ebene, werden die Vatelemente bis hin zum Wurzelement (FSU Jena) dargestellt, während darunter all diejenigen Individuen der Ontologie angezeigt werden, die organisatorisch der aktuellen Ebene untergeordnet sind. In Abbildung 19 sind dies vor allem die Lehrstühle bzw. Professuren des *Institutes für Informatik*. Es werden dem Nutzer nur die nächst folgenden Elemente der Unterebene gezeigt, da eine Darstellung der kompletten Sub-Taxonomie zu viel Platz im Web-Interface beanspruchen würde. Der Nutzer kann, bei Beibehaltung des Suchwortes, in der dargebotenen Taxonomie navigieren, indem er auf den Namen des gewünschten neuen Bereiches klickt.

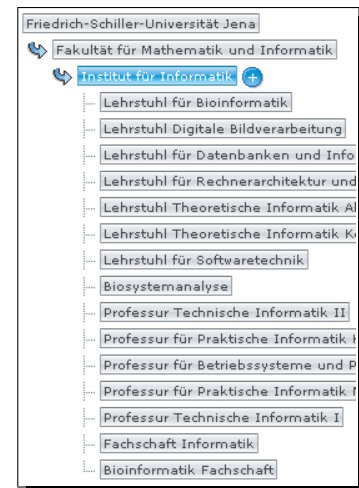


Abbildung 19: Beispiel einer generierten Taxonomie.

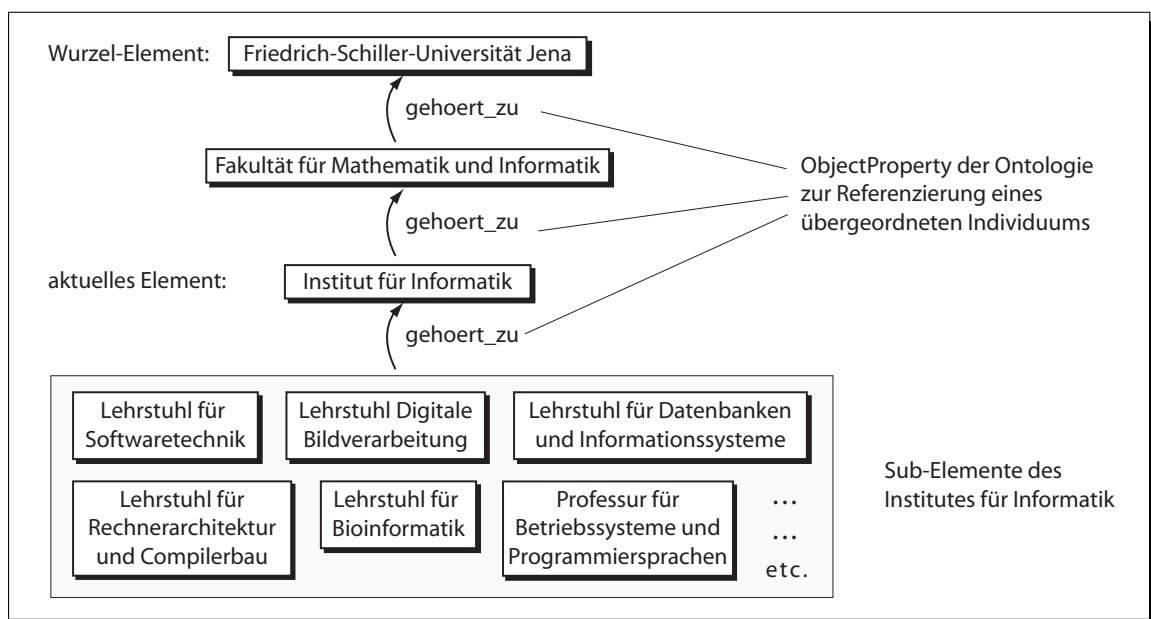


Abbildung 20: Beispiel einer zugrunde liegenden Struktur für eine Taxonomie.

Die Abbildung 20 verdeutlicht, welche Strukturen der gezeigten Beispieltaxonomie (Abbildung 19) zugrunde liegen.

⁸⁵ Der Screenshot in Anhang C auf S. 79, zeigt die Einordnung der Taxonomie im Web-Interface.

⁸⁶ Hier: <http://www.minet.uni-jena.de/www/fakultaet/>. Die Tatsache, dass auch mehrere URLs für ein Individuum zuordenbar sind, soll hier ausgeblendet werden. (Siehe 10.2.)

Umsetzung der Taxonomiegenerierung

Zu jeder Suchraumeinschränkung soll die passende Taxonomie generiert werden. Hierzu wird, ausgehend von der aktuellen Ebene des Suchraumes die Taxonomie sukzessiv bis zum Wurzelement aufgebaut. Genutzt wird hierfür das nach erfolgreichem Parsen der Ontologie zurückgegebene Individuum-Array⁸⁷ des OWLP-Objektes. Die Bestimmung der Sub-Elemente wird ebenfalls durch die Analyse des Individuum-Arrays ermöglicht, dessen Ergebnis der eigentlichen Taxonomie angehangen wird.

Wie Abbildung 20 zeigt, wird die Taxonomie allein durch die Beziehungen, die auf die ObjectProperty *gehört_zu* beruhen, aufgebaut. An dieser Stelle wird die in Abschnitt 8.5 geforderte Notwendigkeit für die Angabe eines Wertes, für die *gehört_zu*-Eigenschaft, deutlich.

Zur Generierung der Taxonomie wird serverseitig in dem Skript *search.php5* die Objekteigenschaft *TaxoHTML* der CONTROL-Klasse aufgerufen. Die Eigenschaft enthält die fertig generierte Taxonomie in einer für das Web-Interface aufbereiteten Form und braucht nur noch an der gewünschten Stelle eingefügt werden.

Für die Generierung der Taxonomie ist die Methode *getTaxo()* der CONTROL-Klasse verantwortlich, die bereits während der Instanzierung einer CONTROL-Objektinstanz aufgerufen wird. Nach Abarbeitung der Methode enthält die Eigenschaftsvariable *TaxoHTML* die Taxonomie in Form eines formatierten Strings.

10.5 Unscharfe Suche zur Suchraumerweiterung

Die Webseitenstruktur der FSU Jena machte es in einigen Fällen notwendig, die gewählte Sucheinschränkung „unscharfer“⁸⁸ zu formulieren und die zuvor gewählte Suchraumeinschränkung teilweise wieder rückgängig zu machen. Ein konkretes Beispiel soll dies näher erläutern:

Befindet sich die *Sonto*^x-Suche in der Taxonomie-Ebene des *Institutes für Informatik* (Abb. 21), so werden nur Webressourcen berücksichtigt, die hinter der zugehörigen URL-Struktur angeordnet sind. Der aktuelle Suchstring lautet z. B.:

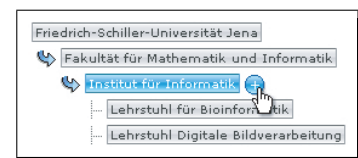


Abbildung 21: Auszug aus der Taxonomie.

Webtechnologien /www/fakultaet site:minet.uni-jena.de OR site:informatik.uni-jena.de

Da viele Bereiche der FSU Jena, darunter auch das Institut für Informatik, keine konsequente Webseitenstruktur besitzen, schlägt die bis jetzt besprochenen Vorgehensweisen

⁸⁷ Siehe Listing 20 auf Seite 54.

⁸⁸ Mit einer *unscharfen* Suche (Fuzzy-Suche) ist der Sachverhalt gemeint, dass nicht mehr unter einer konkreten Organisationsstruktur gesucht wird, sondern über mehrere Strukturen hinweg, so dass es zu ungenauen (unscharfen) Ergebnissen bezüglich der aktuellen Suchraumeinschränkung kommen kann.

fehl. Beispielsweise werden die Webseiten der *Professur für Praktische Informatik (Künstliche Intelligenz)* nicht mit in einer Suche berücksichtigt, da die zugehörige URL⁸⁹ außerhalb der Verzeichnisstruktur des Institutes liegt.

Um diese Art von Sonderfall zu berücksichtigen, wurde in *Sonto*^x die Möglichkeit geschaffen, den gewählten Suchraum mit einem Klick auf ein nebenstehendes Plus-Icon (Abb. 21) „unscharf“ zu erweitern. Die momentane Suchraumeinschränkung wird dadurch fallen gelassen. Der neue Suchstring wird aus dem Suchbegriff, aus dem Individuennamen der aktuellen Ebene und aus den Domänen (*site:-Angaben*) der in der Taxonomie darüber liegenden Ebene gebildet. Die so gewählte Erweiterung der Suche, wird im Web-Interface anhand einer hellblauen Umrahmung der nun „unscharf“ eingeschlossenen Ebenen kenntlich gemacht (Abb. 22). Der Suchstring der mit der Abbildung 22 korrespondiert lautet z. B.:

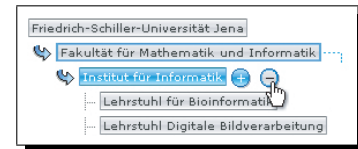


Abbildung 22: Auszug aus der Taxonomie.

Webtechnologien “Institut für Informatik” *site:informatik.uni-jena.de* OR
site:mathematik.uni-jena.de OR *site:minet.uni-jena.de*

Die Webseiten der *Professur für Praktische Informatik (Künstliche Intelligenz)* werden nun bei der Suche mit aufgelistet. Voraussetzung ist hier, dass der String „Institut für Informatik“ an einer Stelle der jeweiligen Webseiten enthalten ist. Ein Klick auf das Minus-Icon hebt die unscharfe Suchraumerweiterung wieder auf.

Gesteuert wird diese Funktionalität in der CONTROL-Klasse über die Methode *fuzzy_control()*, die in der schon erwähnten Methode *getTaxo()* eingesetzt wird und für die jeweilige Modellierung der Taxonomiedarstellung und für das Zusammenstellen der jeweiligen Parameterstrings zu den Verlinkungen der Plus- und Minus-Icons verantwortlich ist.

Diese Art von Suchraumerweiterung stellt einen nicht zufrieden stellenden Ansatz dar und kann keinen positiven Effekt für die Websuche garantieren. Vielmehr wird dem Nutzer damit die Möglichkeit geboten, immer dann, wenn wider Erwarten sehr wenige oder gar keine Treffer für eine Ebene angezeigt werden, den Suchraum etwas zu erweitern. Diese Option *kann* in einigen Fällen den gewünscht Erfolg bringen, während in anderen Situationen kein Effekt damit erzielt wird. Die Option der unscharfen Suche kann daher nur bedingt eine Hilfestellung bieten.

⁸⁹ <http://www.minet.uni-jena.de/~beckstei/>

10.6 Zusatzinformationen zur aktuellen Suchraumeinschränkung

In *Sonto*^x werden rechts neben der Taxonomie, für jede Ebene der Einschränkung des Suchraumes, Zusatzinformationen über die jeweilige ausgewählte Organisationseinheit der FSU Jena angezeigt. Die Informationen selbst beruhen auf den in der Ontologie hinterlegten Zusatzinformationen über das jeweilige aufgenommene Individuum. In Abbildung 23 ist ein Ausschnitt des Web-Interfaces zu sehen, indem die Zusatzinformationen für den aktuellen Suchraum der *Physikalisch Astronomische Fakultät* angezeigt werden.



Abbildung 23: Zusätzliche Informationen zu der aktuellen Suchraumeinschränkung.

Die für eine mögliche Anzeige zur Verfügung stehenden Informationen beruhen auf den zuvor getroffenen Definitionen von Eigenschaften (DatatypProperty) in der Ontologie, wie sie in Tabelle 2 auf Seite 47 aufgeführt wurden. Die Reihenfolge der Auflistung kann in der Datei *config.php*, welche sich im *Sonto*^x-Stammverzeichnis befindet, mit entsprechenden Angaben beeinflusst werden. Die Array-Variable *conf_ShowInfo* bestimmt die jeweilige Reihenfolge. Listing 22 zeigt den entsprechenden Auszug aus der Konfigurationsdatei.

```

1  (...)
2  $conf_ShowInfo = array('Direktor', 'Inhaber', 'Dekan', 'Telefon',
3                        'Fax', 'E-Mail', 'Homepage', 'Anschrift',);
4  (...)

```

Listing 22: Festlegen der Reihenfolge für die Informationsanzeige (*config.php*)

Um die Information anzuzeigen, wird die Methode *getInfo()* der CONTROL-Klasse an entsprechender Stelle aufgerufen. In der Methode selbst wird das Konzept des eingebetteten Frames (*iframe*) genutzt, damit eine eventuelle Ladeverzögerung für die Anzeige des Bildes sich nicht auf die Anzeigegeschwindigkeit des restlichen Web-Interfaces niederschlägt. Hierzu wird in den definierten *iframe* als Quellattribut (*src*) die Datei *info.php5* mit den zuvor in *getInfo()* ermittelten Informationen als Parameter übergeben.

```

1  (...)
2  <iframe src="info.php5?infoElementString=Dekan|Prof. Dr. Paul Seidel||
3      Telefon|03641 947000||Fax|03641 947002||
4      E-Mail|dekanat@paf.uni-jena.de||
5      Homepage|http://www.physik.uni-jena.de||
6      Anschrift|Max-Wien-Platz 1, 07743 Jena||
7      &CurrentTaxoLevel=Physikalisch-Astronomische_Fakultät"
8      frameborder="0" marginheight="0" name="info" width="445">
9  </iframe>
10 (...)

```

Listing 23: Beispiel des Quelltextes für das Einbetten der Anzeige der Zusatzinformationen.

Listing 23 zeigt den entsprechenden Abschnitt des XHTML-Quelltextes, so wie er für das Beispiel letztendlich vom Web-Server an den Browser des Nutzers übertragen wird. Die dabei per HTTP-GET übertragenen Parameter werden in der Datei *info.php5* zur endgültigen

Formatierung der Anzeige ausgewertet und zu einer XHTML-Webseite zusammengefügt. Diese wird dann vom Web-Server an den Nutzer gesendet.

Um die Zeichenkette der Anschrift an den richtigen Stellen umzubrechen, wurden diese Stellen schon während der Wissensakquise mit einem Komma als Separater markiert. Dieses Hilfsmittel erlaubt die Erkennung der richtigen Umbruchstelle und dient einer sauber formatierten Darstellung im Web-Interface. Um unschöne horizontale Scrollbalken bzw. eine unleserliche Anzeige zu vermeiden, sollte dieses Komma bei der Erstellung der Ontologie mit angegeben werden.

Teil III

Auswertung und Zusammenfassung

11 Betrachtung der Umsetzung im Hinblick auf die Problemstellung

Mit Blick auf die Problemstellung soll im Folgenden die durch *Sonto*^x bereitgestellte Web-Suche kritisch betrachtet werden. Es werden dabei die Vorteile, die das System bietet, und die Probleme, die *Sonto*^x momentan nicht lösen kann, getrennt in kompakter Form dargestellt. Dabei werden nur solche Punkte diskutiert, die wesentliche Stärken bzw. Schwächen des Systems darstellen.

11.1 Stärken von *Sonto*^x

Der Einsatz von *Sonto*^x bietet einem Nutzer einige Vorteile im Vergleich zu einer konventionellen Suchmaschine, wie z. B. auf Basis der Google-Suche. Im Folgenden werden die Eigenheiten von *Sonto*^x aufgeführt, die eine Web-Suche auf Basis der Webseiten der FSU Jena erleichtern bzw. erweitern:

- ① Durch die effektive Einschränkung des Suchraumes werden all diejenigen Resultate ausgeblendet, die von vornherein nicht von Interesse sind, da sie nicht unter der Domäne der FSU Jena, bzw. einer Unterstruktur der FSU Jena, angeordnet sind. Ohne dass der Nutzer die genaue Domänenstruktur kennen muss, führt dies zu einer raschen Minimierung der potenziell relevanten Trefferanzahl, mit dem positiven Effekt, dass sehr schnell eine kleine überschaubare Trefferliste für den gesuchten Begriff angezeigt wird.
- ② Dem Nutzer können bereits bei der Auflistung der Suchtreffer, semantische Informationen über die Zugehörigkeit der Webressourcen zu der Universitätsstruktur geboten werden. Ohne die Webseite zu besuchen bzw. genaue Kenntnisse über die Struktur der FSU Jena zu besitzen kann dadurch der Nutzer schnell eine Entscheidung über die jeweilige Relevanz eines Suchtreffers fällen ohne alle Webseiten einzeln zu besuchen.
- ③ Die gebotenen Zusatzinformationen zur aktuellen Suchraumeinschränkung (Bild und nähere Beschreibung), unterstützen zu jeder Zeit der Suche, den Bezug zu der Ebene der FSU Jena, auf welche die Suche beschränkt ist, und vermittelt dadurch einen engeren Bezug zur aktuellen Suchraumeinschränkung.

- ④ Da *Sonto*^X die semantischen Informationen allein aus einer zuvor erstellten Ontologie bezieht, kann durch eine Modifikation der Ontologie eine rasche Anpassung an die jeweiligen Bedürfnisse ermöglicht werden. Diese Datenunabhängigkeit des *Sonto*^X-Systems ermöglicht eine flexibel angepasste Websuche.
- ⑤ Durch die Erstellung einer neuen Ontologie ist es möglich, nicht nur eine lokale, unter einer einzigen Domäne umgesetzte Organisationsstruktur eines Web-Angebotes umzusetzen, sondern darüber hinaus kann in *Sonto*^X auch eine Ontologie von einer virtuellen Organisationsstruktur genutzt werden. Mit einer virtuellen Organisationsstruktur ist z. B. die über das ganze WWW verteilte Menge an unterschiedlichen Domänen gemeint, die sich alle einem gleichen Thema widmen und unter einer hierarchischen Struktur (mit einem Wurzelement) vereinigt sind. *Sonto*^X kann auf Basis dieser neuen Ontologie die einzelnen Quellen zu einer virtuellen Einheit verschmelzen, auf der sich die weitere Websuche beschränkt.

11.2 Grenzen von *Sonto*^X

Die umgesetzte ontologiebasierte Web-Suche zeigt an einigen Stellen Grenzen auf, die hauptsächlich auf der mangelnden Semantik der Webseiten und deren unzureichenden Annotation mit Meta-Daten beruhen. Zu den grundsätzlichen Problemen, die bei einer Suche mit *Sonto*^X auftreten, gehören:

- ① Es können nur solche Webseiten einem Individuum zugeordnet werden, zu denen ein entsprechender Eintrag in der Ontologie vorgenommen wurde. Weil in der Ontologie nur eine Teilmenge der real existierenden Webseitenmenge der Domäne *fsu-jena.de* modelliert wurde, können zwangsläufig Suchtreffer auftreten, für die keine Zuordnung möglich ist. In diesem Fall werden jedoch – wenn möglich – alternativ die Meta-Tags einer HTML-Webseite ausgelesen und angezeigt.
- ② Die Nutzung des Google Web Services garantiert zwar eine umfangreiche Datenbasis, jedoch schlägt sich die Begrenzung der Suchtreffer auf max. zehn pro Anfrage, auf die Umsetzung des gesamten Systems nieder. *Sonto*^X besitzt zu keinem Zeitpunkt der Programmausführung Informationen zu mehr als max. zehn Ressourcen.⁹⁰ Eine eventuelle Neuarrangierung von z. B. 100 Suchtreffern zu einer neuen Liste, beruhend auf einer eigenen Relevanzbewertung, ist somit nicht möglich.
- ③ Das volle Potenzial von *Sonto*^X wird durch die teils mangelnde Umsetzung der Verzeichnisstruktur für die abgelegten Webseiten der einzelnen Bereiche stark ausgebremst. Viele Verzeichnisstrukturen spiegeln nicht die wahre Struktur der Universitätsorganisation wider, wodurch z. B. eine exakte Suchraumeinschränkung auf Basis des *site:-*Konstruktes nicht optimal genutzt werden kann bzw. ganz versagt.
- ④ Der heute noch vorherrschende Mangel an semantischer Annotation der Webseiten, gibt *Sonto*^X nur die Möglichkeit, die Semantik der einzelnen Webressourcen allein

⁹⁰ Optional ist in *Sonto*^X eine Erhöhung auf max. 20 Suchtreffer möglich.

über die IR-Methode der URL-Analyse zu ermitteln. Auf Basis einer Webseitenbewertung ist eine eindeutige Zuordnung zu einem bestimmten Themenbereich nicht möglich.

12 Einsatzszenario des Sonto^X-Systems

An dieser Stelle soll anhand zweier konkreter Suchszenarien die Funktionsweise von Sonto^X dokumentiert werden. Hierzu wird die Verwendung von Sonto^X mit der Suche über **<http://www.google.com>** verglichen. Skizzenhaft wird das Vorgehen und das Resultat beider Varianten gegenübergestellt und abschließend zu jedem Suchszenario eine kurze zusammenfassende Analyse vorgenommen, die die Vorteile des Sonto^X-Systems zur „reinen“ Google-Suche herausstellen.

- ① Es werden Materialien der von der *Fakultät für Mathematik und Informatik* angebotenen Vorlesung *Webtechnologien* gesucht. (Suchbegriff: „Webtechnologien“)
- ② Gesucht sind Informationen über die Vorlesung „Einführung in die Entwicklungspsychologie“ des *Institutes für Psychologie*. (Suchbegriff: „Entwicklungspsychologie“)

Sonto^X-Suche ①

Die 1. Suche ergibt 46 Treffer. Jeder der ersten zehn Treffer zeigt eine gelungene semantische Zuordnung an.
Die 2. Suche wird durch ein Klick auf die Zuordnung *Fakultät für Mathematik und Informatik* verfeinert und liefert 38 Treffer.

Google (www.google.com) ①

Die 1. Suche ergibt über 104 000 Treffer. Die 2. Suche mit zusätzlicher Angabe von *site:uni-jena.de* ergibt angeblich 87 Treffer. Nach 66 Treffern bricht die Auflistung jedoch vorzeitig ab.

Beide Recherchen haben die Anzahl der Treffer mit zwei *Klicks* effektiv auf ein überschaubares Maß verringern können. Bei Google wird jedoch vorausgesetzt, dass der Nutzer mit dem *site*:-Konstrukt vertraut ist. Weiterhin sind bei Sonto^X nur solche Treffer enthalten, die von dem Webangebot der *Fakultät für Mathematik und Informatik* stammen, während bei Google Treffer der gesamten Universität enthalten sind. Bei Sonto^X kann der Nutzer sicher sein, dass alle 38 Treffer aus der angegebenen Fakultät stammen, während bei Google eventuell jeder Treffer manuell überprüft werden muss.

Sonto^X-Suche ②

Die 1. Suche ergibt 768 Treffer, auf die alle zugegriffen werden kann. Auch hier werden passende Zuordnungen gefunden.
Die 2. Suche wird durch ein Klick auf die Zuordnung *Lehrstuhl für Entwicklungspsychologie* verfeinert und liefert 112 Treffer, die eindeutig dem Lehrstuhl zuzuordnen sind.

Google (www.google.com) ②

Die 1. Suche ergibt über 164 000 Treffer. Die 2. Suche mit zusätzlicher Angabe von *site:uni-jena.de* ergibt angeblich 831 Treffer. Nach 268 Treffern bricht auch diese Auflistung vorzeitig ab.

Die Suche mit Sonto^x konnte die potenzielle Trefferliste mit zwei Klicks auf 112 Treffer einschränken, von denen alle zum ausgewählten Lehrstuhl gehören. Google liefert hingegen mit 268 Treffern eine höhere Anzahl, jedoch sind unter den Treffern auch Webressourcen anderer Bereiche der FSU Jena. Der Nutzer müsste daher die Treffer manuell überprüfen. Würde der Nutzer den *site*:-Wert auf die Homepage des *Lehrstuhles für Entwicklungspsychologie* anpassen, würde er ähnliche Ergebnisse wie mit Sonto^x erhalten. Es ist jedoch nicht davon auszugehen, dass ein Nutzer die Webseitenstruktur der FSU Jena im Detail kennt. An dieser Stelle kann Sonto^x seine Stärken ausspielen. Darüber hinaus wird nicht nur Hilfestellung bei der Suchraumeinschränkung gegeben, sondern der Nutzer sieht auf einen Blick, auf welchen Bereich er seine Suche einschränken kann. Zusätzlich werden ihm noch erweiterte Informationen zur ausgewählten Ebene präsentiert, die den semantischen Bezug zur Treffereinschränkung besser deutlich werden lassen.

Beide Suchszenarien können nur einen beschränkten Eindruck vermitteln, welchen Vorteil Sonto^x bei einer Suche bietet. Zusätzlich müssten dafür die semantischen Informationen im Web-Interface und ihre visuelle Aufbereitung mit in Betracht gezogen werden.⁹¹

13 Stellung von Sonto^x in der Semantic Web Vision

Es stellt sich die Frage, wie Sonto^x im Hinblick auf die SW-Vision eingeordnet werden und welchen Beitrag diese Arbeit zur Entwicklung des SWs beisteuern kann.

Sonto^x kann im engeren Sinne nicht als eine SW-Anwendung betrachtet werden, da ein SW momentan noch nicht realisiert ist. Eine Grundvoraussetzung für ein SW ist die Annotation der Webseiten mit semantischen Informationen, wovon im heutigen WWW noch nicht die Rede sein kann. Was jedoch getan werden kann, um trotzdem einen ersten Schritt in Richtung einer SW-Umsetzung zu gehen, ist die Schaffung einer externen Semantikbeschreibung, die heute schon von Anwendungen genutzt werden können. In Sonto^x wurde dies mit der Modellierung einer eigens für das Problem zugeschnittenen Wissensbeschreibung in Form einer Ontologie umgesetzt. Die Ontologie, wenn auch in einem beschränkten Maße, ermöglicht eine semantische Auswertung der entsprechenden Webressourcen im Hinblick auf ihre Einordnung in die Organisationsstruktur der FSU Jena und gibt so ein gutes Beispiel für den Einsatz und Nutzen einer Ontologie im WWW.

In dieser Arbeit wurden die Einsatzmöglichkeiten einer Ontologie für den Spezialbereich einer Websuche untersucht. Die Ergebnisse geben der Entwicklungsgemeinde eventuell neue Denkanstöße und helfen mit, die Akzeptanz und die Verbreitung von Ontologien für eine semantische Auszeichnung im WWW voranzubringen. In diesem Sinn kann Sonto^x als ein kleiner Schritt hin zu einem SW angesehen werden.

Der Versuch einer Prognose über die SW-Entwicklung

Es kann nicht mit Sicherheit vorhergesagt werden, wie und ob sich das SW in der Zukunft umsetzen lässt oder ob es vielleicht nur eine Vision bleiben wird. Doch obwohl es derzeit

⁹¹ Siehe <http://www.artusweb.de/SontoX>

noch viele technische und organisatorische Probleme zu bewältigen gilt, spricht einiges dafür, dass das SW eines Tages Wirklichkeit werden wird.

Die bereits angesprochene „Inflation der Information“ ist ein wichtiger Grund dafür, nach geeigneten Lösungen für die Bewältigung der Informationsflut zu suchen. Mit der Realisierung des SW würde eine Möglichkeit geschaffen werden, dem Computer effizient und effektiv die Arbeit der Informationssuche, -verwaltung, -akquise und -recherche usw. zu überlassen und dem ständig steigenden Informationsangebot entgegenzutreten. Die steigende Nachfrage nach geeigneten Lösungen könnte daher die Entwicklungsbemühungen weltweit forcieren. Dass diese Entwicklung bereits begonnen hat, zeigt die in den letzten Jahren wachsende Anzahl an Publikationen und Forschungsaktivitäten, vor allem an den Hochschulen und auch in der Industrie.

Eine der spannendsten Anwendungsgebiete eines zukünftigen SW ist der Wunsch nach sich autonom im Web bewegenden Agentensystemen (Web-Agents), die selbstständig die Information im Netz durchsuchen und eigenständig Schlüsse ziehen können. In der Literatur finden sich diesbezüglich viele Beispielszenarien.⁹² Diese Beispiele tragen dazu bei, die Idee des SW einem breiten Publikum nahe zu bringen und so die Zahl derer, die sich mit diesem Thema beschäftigen, zu vergrößern. Eine vollständige Etablierung eines SW ist – nach Meinung des Autors – in den kommenden zehn Jahren nicht in Sicht. Jedoch bleibt zu vermuten, dass einige Spezialanwendungen schon bald die Vorteile eines SW klar demonstrieren könnten und so seine Umsetzung – ähnlich der WWW-Entwicklung – beschleunigen wird.⁹³

14 Zusammenfassung

In dieser Arbeit wurde eine ontologiegestützte Websuche entwickelt, die anschaulich demonstriert, welches Potenzial die Nutzung einer Ontologie für eine semantische Suche im WWW bereithält. Als erstes wurde das Problem der Beschaffung einer Datengrundlage durch den Einsatz des Google Web Services erfolgreich gelöst, wobei gleichzeitig ein gutes Beispiel für die Einsatzmöglichkeit von Web Services gegeben wurde. Die Qualität und die Anzahl der von Google indizierten Webressourcen bildete für *Sonto*^x eine wichtige Grundlage, auf die sich die programmierte Websuche stützt und von der auch deren Güte abhängig ist. Als zweite wichtige Komponente wurde eine eigens zugeschnittene Ontologie unter Zuhilfenahme des Ontologieeditors Protégé in der Ontologiesprache OWL erstellt. Ein eigener OWL-Parser übernimmt die Aufgabe der Informationsextraktion aus der Ontologie. Die so erhaltenen Informationen sollten in einer geeigneten Weise mit den Suchtreffern in Verbindung gebracht werden. In *Sonto*^x wurde die Verbindung von Ontologie und Datengrundlage an verschiedenen Stellen erfolgreich umgesetzt und im Web-Interface aufbereitet angezeigt. Ausschlaggebend hierfür ist die OWL-ObjectProperty *gehört_zu*, die eine hierarchische Strukturierung der in der Ontologie umgesetzten Organisationseinheiten erst ermöglicht, und die jedem Individuum zugeordnete OWL-DatatypeProperty *Homepage*, die eine Zuordnung einzelner Suchtreffer zu einem

⁹² Verwiesen sei hier auf Berners-Lee Beschreibung eines SW-Anwendungsszenarios für Web-Agenten in [BHL01].

⁹³ Die Entwicklung von *Sonto*^x trägt ihren Teil dazu bei.

Individuum sicherstellt. Die dafür nötige Programmierung der zugrunde liegenden Steuerlogik gestaltete sich an vielen Stellen als sehr komplex und aufwändig. Die schlussendlich im Web-Interface dargebotene Suche mit den zusätzlichen semantischen Auszeichnungen lassen kaum erahnen, dass allein die Klassenbibliothek von *Sonto*^x mehr als 1800 Zeilen PHP-Quelltext umfasst.⁹⁴ Am problematischsten zeigten sich die vielen Spezialfälle, die meist aus einer ungeeigneten und unsauberen Gestaltung der Webseitenstruktur der FSU Jena resultierten und eine teils aufwändige Behandlung erforderten.

Mit dem *Sonto*^x-System wird eine effektive Websuche auf den Webseiten der FSU Jena ermöglicht, die einen Nutzer schnell zum Ziel führt. Jedoch bietet das System nicht zu allen Suchanfragen Vorteile bezüglich einer „normalen“ Websuche. *Sonto*^x stellt jedoch eine interessante neue Art einer Websuche dar, die im Sinne der SW-Vision ein Beispiel für einen erfolgreichen Einsatz einer Ontologie für eine semantische Auswertung demonstriert. Mit *Sonto*^x ist, nach Wissen des Autors, eine bis jetzt im Web noch nicht existierende Verbindung zwischen einer Ontologie und einer Suchmaschine gelungen.

15 Ausblick

Es bleibt abzuwarten, ob *Sonto*^x sich im Praxiseinsatz bewährt und von der Nutzergemeinde angenommen wird. Vielleicht bewirkt *Sonto*^x ein Umdenken einiger Webmaster, die dadurch erkennen, dass ihre Seiten schlecht annotiert sind oder ihre gewählte Verzeichnisstruktur eine Websuche erschweren bzw. verhindern. Auf Basis einer sauberen Domänen- und Verzeichnisstruktur könnte *Sonto*^x sein Potenzial voll ausspielen.

Da momentan der Google Web Service als Methode für die Datenbeschaffung eingesetzt wird, ist die Zukunft von *Sonto*^x abhängig von dem sich offiziell immer noch im Beta-Status befindlichen Web Service. Es ist jedoch anzunehmen, dass Google seinen Dienst aufrechterhalten wird. Falls nicht, kann als alternative Methode das Screen Scraping verwendet werden.

Ideen für eine Weiterentwicklung der Websuche sieht der Autor noch viele. So könnte z. B. dem Nutzer auf der Optionsseite die Möglichkeit zur Angabe seiner eigenen Ontologie eingeräumt werden. Weiterhin liegt in der modularen Architektur weiteres Potenzial, die Beschaffung der Datengrundlage auf eine alternative Suchmaschine aufzubauen.

Den größten Nutzen könnte *Sonto*^x jedoch dadurch erreichen, dass auf Basis unterschiedlicher Ontologien jeweils unterschiedliche Suchräume abgedeckt werden. So wäre es z. B. möglich, Domänen-übergreifende Organisationsstrukturen in einer Ontologie abzubilden und sie dann in *Sonto*^x einzubinden. Auf diese Weise könnte das Suchverhalten je nach Wunsch variiert und angepasst werden. So eingesetzt, stellt *Sonto*^x ein universelles Werkzeug für eine erweiterte Websuche dar, wobei die Treffer mit den jeweiligen semantischen Informationen aus der Ontologie aufbereitet werden.

⁹⁴Wird die zur Etablierung eines SOAP-Clients verwendete NuSOAP-Klasse einberechnet, so besteht die gesamte *Sonto*^x-Klassenbibliothek sogar aus ca. 5 800 Zeilen Quelltext.

A Glossar

Agent: Oder Web-Agent, ist ein Softwareprogramm, welches vom Nutzer beauftragt, völlig autonom das WWW nach relevanten Informationen absucht, wobei ein Web-Agent auch Teilaufgaben an andere Web-Agenten abgeben kann. Grundvoraussetzung ist ein etabliertes semantisches Web.

API: (Application Program Interface) Bezeichnet eine Programmschnittstelle, die es einen Softwareentwickler erlaubt, Funktionalitäten eines Programms in eine eigene Anwendung zu integrieren.

Browser: Ein spezielles Programm, das auf dem Computer des WWW-Nutzers läuft und welches die in HTML kodierten Webdokumente in eine am Monitor darstellbare Form überführt.

Client: Bezeichnet ein Programm, welches einen Server kontaktiert und von diesem Informationen anfordert. Der im WWW eingesetzte Browser ist in diesem Sinne ein Client. Aber es gibt auch andere Clients im WWW, die WWW-Server kontaktieren und Informationen von diesen herunterladen, wie z.B. Suchmaschinen oder Agenten.

DNS: (Domain Name Service) Gehört zu den Verzeichnisdiensten im Internet. Der Dienst ermöglicht eine Zuordnung logischer Bezeichnungen zu einer numerischen IP-Adresse. Diese Bezeichnung erleichtert Menschen einen erheblich leichteren Umgang mit den unterschiedlichen Netzwerk-Endsystemen, als die Verwendung bloßer IP-Adressen. Die Rückübersetzung einer IP-Adresse aus einer logischen Bezeichnung übernehmen die im Netz verteilten DNS Name-Server.

HTML: (Hypertext Markup Language) Das einheitliche Dokumentenformat für Hypermedia-Dokumente im WWW. Dokumente, die im WWW übertragen und vom Browser dargestellt werden sollen, sind in HTML kodiert.

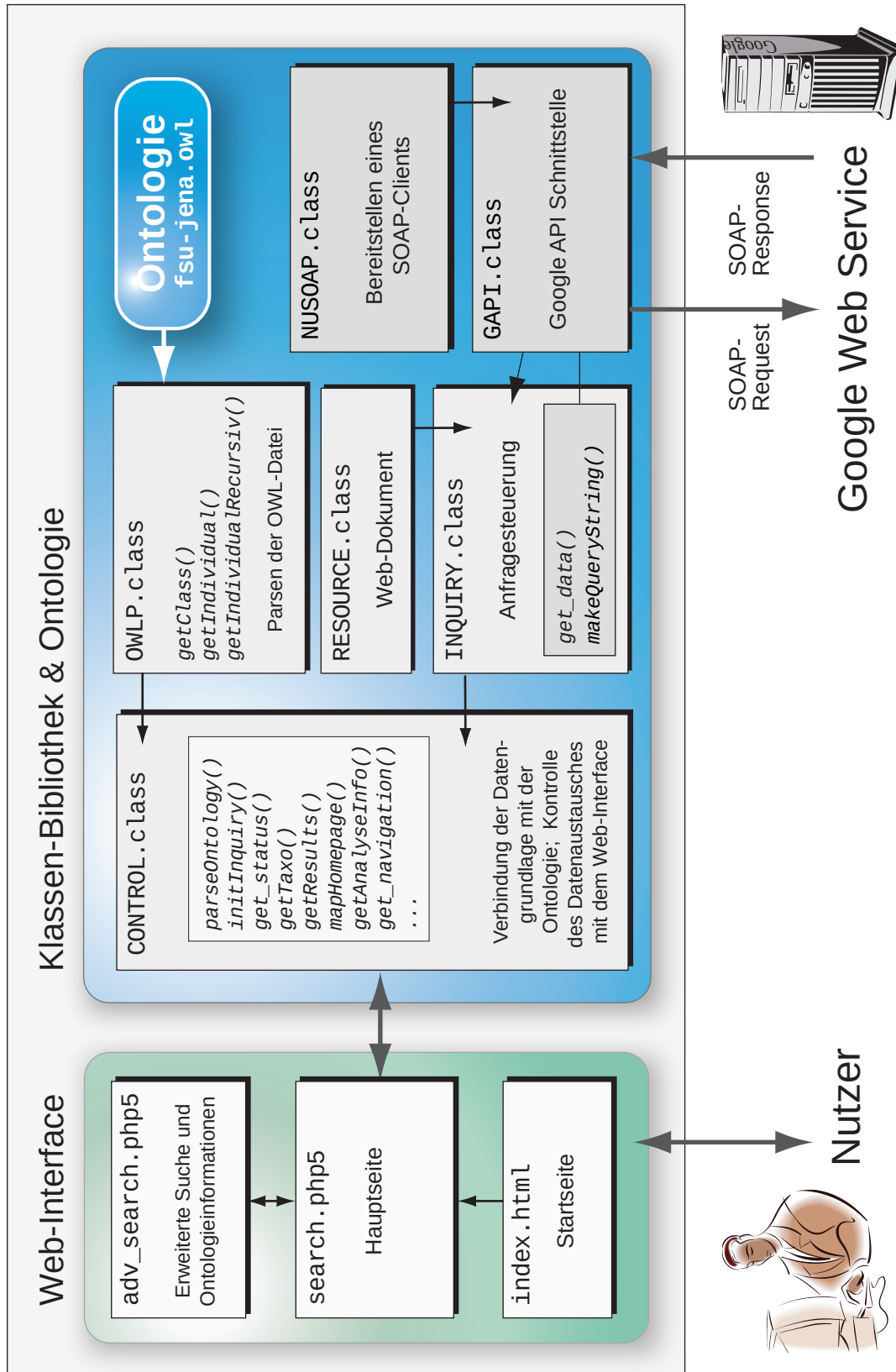
HTTP: (Hypertext Transfer Protocol) Das Protokoll, das die Kommunikation von Browsern und WWW-Servern im WWW regelt. Fordert ein Browser ein Dokument vom WWW-Server an oder beantwortet der WWW-Server eine Anfrage, muss diese Anfrage den Konventionen des HTTP-Protokolls gehorchen.

Information-Retrieval: (IR) Bezeichnet die Methode der computergestützten inhaltsorientierten Suche in einer Menge von Dokumenten bzw. Datenbeständen zur Informationsgewinnung. Dabei liegt das Ziel darin, die implizit in den Datenbeständen enthaltenen Informationen zu extrahieren. Im Zusammenhang mit einer Informationsgewinnung auf Basis des WWW wird hier auch von Online-Retrieval gesprochen.

Namensdienst: (Naming Service), ein im Netzwerk implementierter Mechanismus, der logische, leicht merkbare Namen einer Ressource oder einer Person auf numerische Netzwerkadressen abbildet.

- Ontologie:** Im Kontext des Semantic Web wird darunter eine formale Sammlung und Strukturierung zusammengehöriger Begriffe verstanden. Die in einer Ontologie zusammengeführten Begriffe werden in ihr geordnet, hierarchisiert und miteinander in definierte Beziehungen gebracht. Das Wissensgebiet, das mit Hilfe einer Ontologie beschrieben und erschlossen wird, wird hierbei als Domäne bezeichnet.
- OWL:** (Web Ontology Language) Ist eine semantische Auszeichnungssprache (Markup-Sprache) zum Veröffentlichen und Austauschen von Ontologien im WWW. Die Sprache ist eine Weiterentwicklung der Ontologie-Sprache DAML+OIL
- PHP:** (PHP Hypertext Preprocessor) Ist eine serverseitig, in HTML eingebettete Webskriptsprache, mit deren Hilfe eine schnelle und effiziente Entwicklung dynamischer Webanwendungen ermöglicht wird.
- Parser:** Ein Programm, das ein Dokument auf Basis einer speziellen Spezifikation auf syntaktische/semantische Korrektheit hin überprüft bzw. zusätzlich die in dem Dokument strukturiert abgelegten Informationen für eine Weiterverarbeitung interpretiert.
- Semantic Web:** Ist eine Erweiterung des gegenwärtigen WWW um eine wohl definierte Bedeutung der Informationen. Es ist eine Initiative des W3C und einer großen Zahl von Interessenten aus Forschung und Industrie. Es basiert auf XML und RDF als Sprachsyntax und URIs zur eindeutigen Identifizierung von Dokumenten bzw. Objekten. Darauf soll eine Vielzahl von Applikationen aufbauen.
- Server:** Bezeichnet einen Prozess, der von Clients kontaktiert wird, um diesen Informationen zurück zu liefern. Oft wird auch der Rechner, auf dem ein Server-Prozess abläuft als Server bezeichnet.
- SOAP:** Bei SOAP handelt es sich um ein von IBM aus XML-RPC abgeleitetes auf XML basierendes Protokoll zur Nachrichten-Übermittlung. Das Protokoll regelt speziell die Kommunikation zwischen Maschine und Maschine und bildet somit einen wichtigen Bestandteil der Web Services Architektur. SOAP ist seit der vom W3C verabschiedeten Version 1.2 kein Akronym mehr, sondern steht nur noch für sich selbst.
- URI:** (Uniform Resource Identifier) Eine allgemeine Form des URL (Uniform Resource Locator). Durch die Angabe eines URI kann eine Ressource eindeutig referenziert werden.
- Web Service:** Web Services sind spezielle Dienste, welche eine Kommunikation zwischen Maschinen erlauben. Der Dienstanbieter (der Server) stellt einen speziellen Dienst (Service) für andere Rechner (die Clients) über das Web zur Verfügung. Die Idee basiert auf dem Server-Client-Prinzip, wobei der Client eine speziell formulierte Anfrage an den Server stellt, welcher in Abhängigkeit seines angebotenen Dienstes die Anfrage bearbeitet und das Ergebnis an den Client zurück sendet.

B Das Architektur-Modell von Sonto^x



C Screenshots des Sonto^X Web-Interfaces



Abbildung 24: Web-Interface: Startseite (<http://www.artusweb.de/SontoX/index.html>)

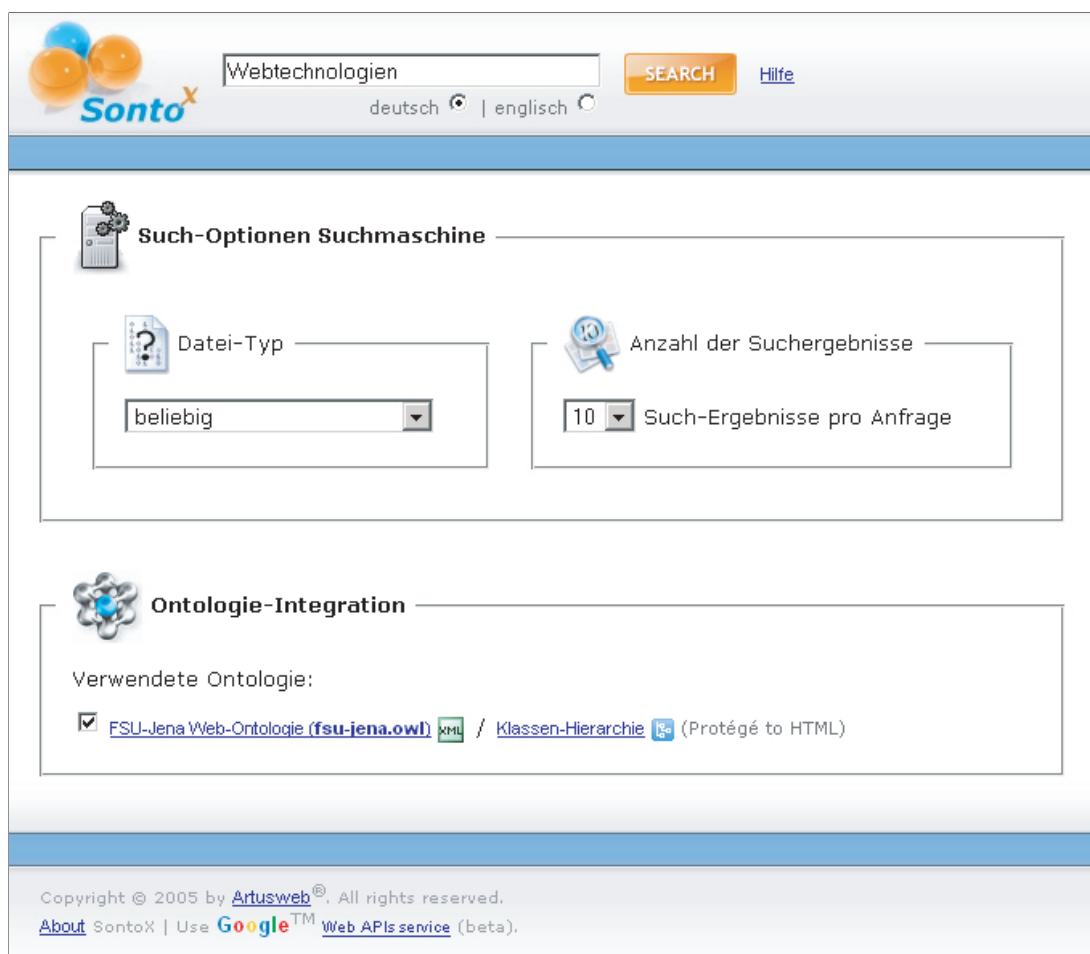


Abbildung 25: Web-Interface: Erweiterte Einstellungen ([adv_search.php5](#))

The screenshot displays the SontoX web interface. At the top, there is a search bar with the query 'Webtechnologien' and a 'SEARCH' button. To the right of the search bar, there are links for 'erweiterte Suche' and 'Hilfe'. Below the search bar, there is a language selector showing 'deutsch' and 'englisch'. On the right side of the top bar, there is a Google logo and a 'Web APIs service' badge showing '0.089 sec.' and '77/1000'.

Below the top bar, there is a navigation menu for the 'Friedrich-Schiller-Universität Jena'. The menu items are: 'Chemisch-Geowissenschaftliche Fakultät', 'Biologisch-Pharmazeutische Fakultät', 'Fakultät für Mathematik und Informatik' (highlighted with a mouse cursor), 'Philosophische Fakultät', 'Medizinische Fakultät', 'Physikalisch-Astronomische Fakultät', 'Fakultät für Sozial- und Verhaltenswissenschaften', 'Rechtswissenschaftliche Fakultät', 'Wirtschaftswissenschaftliche Fakultät', 'Theologische Fakultät', 'Universitätsrechenzentrum', and 'Digital Teaching Workspace'.

Below the navigation menu, there is a search result summary showing '1 - 10 von insgesamt 47 Treffern'. The search results are listed in a table with two columns: the search result and the source. The results are:

- Webtechnologien**
Anfang Zurück Weiter Ende Index Homepage Text. Folie 29 von 34. Klinisches Rechenzentrum Bachstraße. Internet UIniversitätsklinikum Jane ...
http://www.med.uni-jena.de/wzi/fp2000_praes/sld029.htm 6k
Klinikum der Friedrich-Schiller-Universität Jena
Medizinische Fakultät
- Erweiterte Webtechnologien**
Anfang Zurück Weiter Ende Index Homepage Text. Folie 37 von 46. Klinisches Rechenzentrum Bachstraße. Internet UIniversitätsklinikum Jane ...
<http://www.med.uni-jena.de/wzi/internet2000/sld037.htm> 7k
Klinikum der Friedrich-Schiller-Universität Jena
Medizinische Fakultät
- Vorlesung: Webtechnologien**
Einführung, Themenübersicht und Materialien zur Vorlesung.
<http://www.informatik.uni-jena.de/~sack/WS0405/webtechnologien.htm> 9k
Fakultät für Mathematik und Informatik
- Vorlesung: Webtechnologien - Materialien**
Einführung, Themenübersicht und Materialien zur Vorlesung.
<http://www.informatik.uni-jena.de/~sack/WS0405/webtechnologien-materialien.htm>
Fakultät für Mathematik und Informatik
- Webtechnologien**
Ziel dieser Vorlesung ist, die ...
<http://www.informatik.uni-jena.de/sack/WS0405/webtechnologien.htm>.
Dr.rer.nat. Harald Sack, Institut für Informatik, FSU Jena, ...
<http://www.minet.uni-jena.de/~sack/WS0405/materialien/webtechnologien-01-prt.pdf>
Fakultät für Mathematik und Informatik
- Microsoft PowerPoint - webtechnologien-04.ppt**
<http://www.informatik.uni-jena.de/sack/WS0405/webtechnologien.htm> ...
Webtechnologien -
Vorlesungsinhalt. Teil I: Internet und WWW ...
<http://www.minet.uni-jena.de/~sack/WS0405/materialien/webtechnologien-04sm.pdf>
Fakultät für Mathematik und Informatik
- Microsoft PowerPoint - webtechnologien-06.ppt**
Wintersemester 2004/2005. Webtechnologien. Webtechnologien.
<http://www.informatik.uni-jena.de/sack/WS0405/webtechnologien.htm>.
Webtechnologien.
Dr.rer.nat. ...
<http://www.informatik.uni-jena.de/~sack/WS0405/materialien/webtechnologien-06.pdf>
Fakultät für Mathematik und Informatik
- Microsoft PowerPoint - webtechnologien-06.ppt**
<http://www.informatik.uni-jena.de/sack/WS0405/webtechnologien.htm>.
Webtechnologien.
Dr.rer.nat. Harald Sack, Institut für Informatik, FSU Jena, ...
<http://www.minet.uni-jena.de/~sack/WS0405/materialien/webtechnologien-06sm.pdf>
Fakultät für Mathematik und Informatik

At the bottom of the page, there is a footer showing 'Seite: [1] 2 3 4' and a copyright notice: 'Copyright © 2005 by Artusweb®. All rights reserved. About SontoX | Use Google™ Web APIs service (beta).'

Abbildung 26: Web-Interface: Ergebnis einer Beispielanfrage (search.php5)

Literaturverzeichnis

- [ABK+02] R. Anderson, M. Birbeck, M. Kay, u.a.: *XML professionell*, MITP-Verlag, Bonn, 2002.
- [AH04] G. Antoniou, F. Harmelen: *A Semantic Web Primer*, The MIT Press, Cambridge, Massachusetts, 2004.
- [Bab01] U. Babiak: *Effektive Suche im Internet*, O'Reilly Verlag, Köln, 2001.
- [Ber01] M. K. Bergman: *The deep Web: Surfacing Hidden Value* Journal of Electronic Publishing from the University of Michigan, July 2001, <http://www.brightplanet.com/technology/deepweb.asp>
- [Ber99] T. Berners-Lee: *Weaving the Web – the original design and ultimate destiny of the World Wide Web*, HarperCollins Publishers, Inc., New York, 1999.
- [Ber98] T. Berners-Lee: *Semantic Web Road map – A road map for the future, an architectural plan untested by anything except thought experiments.*, <http://www.w3.org/DesignIssues/Semantic.html>, 1998.
- [BHL01] T. Berners-Lee, J. Hendler, O. Lassila: *The Semantic Web*, Scientific American 284(5), 2001, 34–43.
- [BP98] S. Brin, L. Page: *The Anatomy of a Large-Scale Hypertextual Web Search Engine*. 1998, <http://www-db.stanford.edu/pub/papers/google.pdf>
- [DJ04] W. Dostal, M. Jeckle: *Semantik und Web Services*, in: JavaSPEKTRUM, 4/2004, <http://www.javaspektrum.de>
- [Fer03] R. Ferber: *Information Retrieval – Suchmodelle und Data-Mining-Verfahren für Textsammlungen und das Web*, dpunkt.verlag GmbH, Heidelberg, 2003.
- [FHLW03] D. Fensel, J. Hendler, H. Lieberman, W. Wahlster u.a.: *Spinning the Semantic Web – Bringing the World Wide Web to its Full Potential*, The MIT Press, Cambridge, Massachusetts, 2003.
- [Gil01] W. J. Gilmore: *PHP professionell – Das Handbuch für Umsteiger und Fortgeschrittene*, Galileo Press GmbH, Bonn, 2001.
- [Glö03] M. Glöggler: *Suchmaschinen im Internet – Funktionsweise, Ranking, Methoden, Top Positionen*, Springer-Verlag, Berlin Heidelberg, 2003.
- [Google] Google – Google Web APIs Reference: <http://www.google.com/apis/reference.html>
- [GS05] A. Gulli, A. Signorini: *The Indexable Web is More than 11.5 billion pages:*, in: WWW 2005, May 2005, Chiba, Japan. ACM 1595930515/05/0005, <http://www.cs.uiowa.edu/~assignori/web-size/size-indexable-web.pdf>

- [HL04] T. Hauser, U. M. Löwer: *Web Services – Die Standards*, Galileo Press GmbH, Bonn, 2004.
- [Hes02] W. Hesse: *Ontologie(n)*, Informatik Spektrum, Springer-Verlag, Berlin, Heidelberg, 2002.
- [Jec04] M. Jeckle: *Web Services – Grundlegende Informationen zum im Entstehen begriffenen Technikgebiet Web Services*, 2004, <http://www.jeckle.de/webServices/index.html>
- [Kra04] J. Krause: *PHP 5 – Grundlagen und Profiwissen – Webserver-Programmierung unter Windows und Linux*, Carl Hanser Verlag, München, 2004.
- [MS04] C. Meinel, H. Sack: *WWW – Kommunikation, Internetworking, Web-Technologien*, Springer, Berlin, 2004.
- [OWL] Web Ontology Working Group (W3C): *Web Ontology Language (OWL)*, <http://www.w3.org/2004/OWL/>
- [SOAP] World Wide Web Consortium – SOAP Messaging Framework: <http://www.w3.org/2003/REC-soap12-part1-20030624/>
- [Tie03] S. Tietz: *Kurzstudie – „Software zum Ontologiemangement mit OWL“*, Freie Universität Berlin & Humboldt-Universität zu Berlin, Berlin, 2003. <http://141.20.27.87/webportal/reports/Software%20zum%20Ontologiemangement.pdf>
- [UDDI] Organisation for the Advancement of Structured Information Standards – UDDI Spezifikation: <http://www.oasis.open.org/committees/uddi-spec/doc/tcspecs.htm>
- [W3Ca] The World Wide Web Commity – W3C: <http://www.w3c.org/>
- [W3Cb] World Wide Web Consortium – Semantic Web Activity des W3C: <http://www.w3.org/2001/sw/>
- [W3Cc] World Wide Web Consortium – RDF Vocabulary Description Language 1.0: RDF Schema, 2003, <http://www.w3.org/TR/rdf-schema>
- [W3Cd] World Wide Web Consortium – Web Service Architecture Working Group: <http://www.w3.org/2002/ws/>

- [W3Ce] World Wide Web Consortium – Resource Description Framework:
<http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>
- [WB04] D. Westphal, C. Bizer: *Introduction to RAP – RDF API for PHP V0.9.1*, 2004, *<http://www.wiwiss.fu-berlin/suhl/bizer/rdfapi/tutorial/intruductionToRAP.htm>*
- [Wei01] J. Weizenbaum: *Joseph Weizenbaum – Computermacht und Gesellschaft*, Suhrkamp Verlag, Frankfurt am Main, 2001.
- [WSDL] World Wide Web Consortium – Web Services Description Language (WSDL) 1.1, 2001, *<http://www.w3.org/TR/wsdl>*

(Stand der Webressourcen: Juli 2005)

Ehrenwörtliche Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig und ohne unerlaubte Hilfe Dritter verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Diese Arbeit lag in gleicher Weise noch keiner Prüfungsbehörde vor und wurde bisher noch nicht veröffentlicht.

Jena, den 21. Juli 2005

.....